TUGAS AKHIR – KI141502

IMPLEMENTASI POX PADA PERANGKAT LUNAK SOFTWARE-DEFINED NETWORKING CONTROLLER UNTUK DATA CENTER BERBASIS CONTAINER

DHANAR PRAYOGA
NRP 5113100049

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D

Dosen Pembimbing II
Ir. Muchammad Husni, M.Kom.

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

IMPLEMENTASI POX PADA PERANGKAT LUNAK SOFTWARE-DEFINED NETWORKING CONTROLLER UNTUK DATA CENTER BERBASIS CONTAINER

DHANAR PRAYOGA
NRP 5113100049

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D

Dosen Pembimbing II
Ir. Muchammad Husni, M.Kom.

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – KI141502

POX IMPLEMENTATION ON SOFTWARE-DEFINED NETWORKING CONTROLLER SOFTWARE FOR CONTAINER BASED DATA CENTER

DHANAR PRAYOGA
NRP 5113100049

First Advisor
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D

Second Advisor
Ir. Muchammad Husni, M.Kom.

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

IMPLEMENTASI POX PADA PERANGKAT LUNAK SOFTWARE-DEFINED NETWORKING CONTROLLER UNTUK DATA CENTER BERBASIS CONTAINER

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
DHANAR PRAYOGA
NRP : 5113100049

Disetujui oleh Pembimbing Tugas Akhir:

1. Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.
   NIP: 19770824 200604 1 001 ..... (Pembimbing 1)
2. Ir. Muchammad Husni, M.Kom.
   NIP: 19600221 198403 1 001 ..... (Pembimbing 2)

SURABAYA
JUNI, 2017

(Halaman ini sengaja dikosongkan)

IMPLEMENTASI POX PADA PERANGKAT LUNAK SOFTWARE-DEFINED NETWORKING CONTROLLER UNTUK DATA CENTER BERBASIS CONTAINER

Nama Mahasiswa : DHANAR PRAYOGA

NRP : 5113100049

Jurusan : Teknik Informatika FTIF-ITS

Dosen Pembimbing 1 : Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.

Dosen Pembimbing 2 : Ir. Muchammad Husni, M.Kom.

ABSTRAK

Virtual data center adalah implementasi virtualisasi untuk sebuah data center. Virtualisasi data center dilakukan dengan memvirtualisasikan seluruh perangkat dan jaringan yang ada pada sebuah data center. Software container dapat menggantikan peran virtual host, sehingga memungkinkan implementasi virtual data center berbasis software container. Dalam implementasi virtual data center, dibutuhkan pengaturan jaringan yang fleksibel mengingat siklus hidup mesin virtual yang pendek sehingga membuat bentuk jaringan yang selalu berubah.

Software Defined Networking (SDN) adalah sebuah arsitektur jaringan yang memudahkan Network Administrator untuk mengatur jaringan pada lingkungan virtual data center. Dalam mengimplementasikan SDN, dibutuhkan SDN Controller dan switch yang mendukung OpenFlow. Tugas akhir ini mengimplementasikan POX untuk membangun SDN Controller

serta membuat jaringan virtual dan fisik yang mendukung OpenFlow.

Tugas akhir ini menunjukkan bahwa POX dapat digunakan untuk membangun SDN Controller untuk sebuah virtual data center. SDN Controller yang dibangun dapat mengatasi masalah jaringan virtual pada virtual data center, seperti jaringan yang memiliki looping dan bentuk jaringan yang selalu berubah. SDN Controller yang dibangun juga dapat digunakan untuk mengatur jaringan yang menggunakan switch fisik MikroTik.

Kata kunci: Software Defined Networking, POX, Virtual Data Center, Software Container.

POX IMPLEMENTATION ON SOFTWARE-DEFINED NETWORKING CONTROLLER SOFTWARE FOR CONTAINER BASED DATA CENTER

Student's Name	**: DHANAR PRAYOGA**
Student's ID	**: 5113100049**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.**
Second Advisor	**: Ir. Muchammad Husni, M.Kom.**

ABSTRACT

Virtual data center is an implementation of data center virtualization. Data center virtualization is done by virtualizing all devices and network inside a data center. Software container can replace the use of virtual host, enabling an implementation of container based virtual data center. In implementing virtual data center, there is a need for a flexible network setting, considering the short life cycle of a virtual machine, causing the shape of the network to change ceaselessly.

Software Defined Networking (SDN) is a network architecture that ease the Network Administrator job in administrating the network in a virtual data center environment. In implementing SDN, an SDN Controller and OpenFlow Supporting switches is needed. This Undergraduate Thesis implements POX to develop SDN Controller and develops virtual and physical network that supports OpenFlow.

This Undergraduate Thesis shows that POX can be used to develop SDN Controller for a virtual data center. The SDN Controller can overcome the problems of a virtual network in a virtual data center, i.e. a loop in the network and a dynamic network shape. The SDN Controller can also be used to control the network consisting physical MikroTik switches.

Keywords : Software Defined Networking, POX, Virtual Data Center, Software Container.

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah Swt yang telah melimpahkan rahmat dan hidayah-Nya dan junjungan Nabi Muhammad SAW sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"**IMPLEMENTASI POX PADA PERANGKAT LUNAK SOFTWARE-DEFINED NETWOKING CONTROLLER UNTUK DATA CENTER BERBASIS CONTAINER**"

yang merupakan salah satu syarat dalam menempuh ujian sidang guna memperoleh gelar Sarjana Komputer. Selesainya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Ibu Dina Meinarsari dan Bapak Bagus Haryosuseno selaku orang tua penulis yang selalu memberikan dukungan doa, moral, dan material yang tak terhingga kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Bapak Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.D selaku pembimbing I yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
3. Ir. Muchammad Husni, M.Kom. selaku pembimbing II yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
4. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala jurusan Teknik Informatika ITS.
5. Bapak Dr. Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator S1 di Jurusan Teknik Informatika ITS.

6. Seluruh dosen dan karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di ITS.
7. Sesha Ayu Permatasari dan Bintang Pria Prasetya selaku saudara yang selalu mendukung dalam pengerjaan Tugas Akhir ini.
8. Teman-teman seperjuangan RMK NCC/KBJ, yang telah menemani dan menyemangati penulis.
9. Teman-teman Teknik Informatika ITS angkatan 2013, yang sudah mendukung saya selama perkuliahan.
10. Muhammad Fikri Alauddin sebagai *Comrade* yang selalu mengajak dan membantu penulis untuk menyelesaikan Tugas Akhir ini.
11. Adhika Putra Wicaksono, Annas Abdurrahman Asmiragani, Habibur Rohman, Haris Swastikoputra, Yudhiantono Atidhira, Irtanty Ocvitasari, Danissa Hanum, Firza Sharfina Izzati, Nabilla Fadlina, Namira Nurmalatya, Nastiti Imana, Muhammad Bagas, Teuku Nuraksatra, dan teman teman Sulastri lainnya yang selalu menghibur penulis dalam pengerjaan Tugas Akhir.
12. Sahabat penulis yang tidak dapat disebutkan satu per satu yang selalu membantu, menghibur, menjadi tempat bertukar ilmu dan berjuang bersama-sama penulis.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan sehingga dengan kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Mei 2017

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

DAFTAR PSEUDOCODE

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi yang terjadi dewasa ini menunjukkan tren yang mengarah pada virtualisasi. Virtualisasi yang terjadi tidak hanya pada virtualisasi perangkat, namun juga virtualisasi jaringan (NV) dan virtualisasi fungsi jaringan (NFV) Virtualisasi memiliki banyak keuntungan dari berbagai sisi. Keuntungan, terutama dalam sisi ekonomi, membuat pasar memilih untuk memvirtualisasikan perangkat mereka ketimbang membeli dan memelihara perangkat secara tersendiri. Virtualisasi juga memiliki banyak permasalahan. Salah satu permasalahan yang dihadapi adalah permasalahan konfigurasi jaringan yang harus dilakukan setiap kali melakukan *deploy* perangkat *virtual*. Konfigurasi jaringan yang selalu berubah ini tidak memungkinkan network administrator untuk merubah konfigurasi jaringan setiap ada perangkat virtual baru, sehingga dapat disimpulkan dibutuhkan sebuah solusi yang mampu menjawab permasalahan ini.

Software-Defined Networking (SDN)[1] dapat menyelesaikan permasalahan yang dialami pada *virtual data center*. SDN adalah sebuah arsitektur jaringan yang memisahkan *data plane* dan *control plane*. Hal ini memungkinkan *Network Administrator* melakukan konfigurasi jaringan secara otomatis dan hanya mengakses sebuah SDN *Controller*, sehingga memudahkan *Network Administrator* dalam mengelola jaringan.

POX[2] adalah sebuah *platform* berbasis bahasa pemrograman Python yang digunakan untuk mengembangkan perangkat lunak SDN *Controller*. POX memiliki beberapa komponen yang dapat digunakan ulang untuk membuat SDN *Controller* sesuai dengan kebutuhan pengguna. Hal ini memungkinkan untuk pembuatan perangkat lunak SDN *Controller*

yang cocok untuk sebuah jaringan spesifik, contohnya *virtual data center* berbasis *Software Container.*

Tugas Akhir ini menawarkan sebuah solusi pembangunan *SDN* Controller menggunakan teknologi POX yang akan berjalan diatas jaringan *virtual* berbasis Mininet[3]. Penulis berharap perangkat lunak *SDN Controller* dapat menyelesaikan permasalahan konfigurasi jaringan yang dihadapi. Sehingga dapat membantu *Network Administrator* untuk melakukan konfigurasi jaringan yang optimal guna tercapainya optimasi virtualisasi perangkat.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, dapat dirumuskan beberapa permasalahan sebagai berikut :

1. Bagaimana membuat sistem jaringan *virtual* berbasis SDN yang dapat memanfaatkan SDN Controller yang dibangun menggunakan *platform* POX?
2. Bagaimana mengimplementasikan sistem *Virtual Data Center* berbasis *Software Container* pada jaringan *virtual*(NV)?
3. Bagaimana mengimplementasikan POX untuk pembuatan *SDN Controller* pada jaringan *virtual* (NV) Mininet?
4. Bagaimana menghubungkan *SDN Controller* berbasis POX dengan *Virtual Data Center* berbasis *Software Container*.
5. Bagaimana mengevaluasi performa sistem yang dibangun pada Tugas Akhir.

1.3 Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan masalah sebagai berikut :

1. Menggunakan *platform* virtualisasi Mininet diatas sistem operasi Linux.
2. Menggunakan bahasa pemrograman Python dan platform POX untuk pembuatan *Software-Defined Networking Controller*.

1.4 Tujuan

Tujuan dari pengerjaan Tugas Akhir ini adalah :

1. Membuat SDN Controller menggunakan Bahasa Python dengan *platform* POX.
2. Membuat API pembangunan jaringan *virtual* yang mampu menggunakan SDN Controller sebagai *Controller* jaringannya.
3. Melakukan pengujian dan performansi dari sistem yang dibangun pada tugas akhir ini.

1.5 Manfaat

Pengerjaan tugas akhir ini memiliki manfaat mempermudah *Network Administrator* mengkonfigurasi jaringan pada lingkungan virtualisasi yang memiliki konfigurasi berubah-ubah.

Pengerjaan tugas akhir ini juga memiliki manfaat tersendiri bagi penulis. Manfaat bagi penulis yaitu sebagai sarana untuk mengimplementasikan telah dipelajari selama kuliah agar berguna bagi masyarakat.

1.6 Metodologi

Berikut ini metodologi yang diterapkan dalam pengerjaan tugas akhir :

1.6.1 Penyusunan Proposal

Tahap awal pengerjaan tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dikerjakan.

Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pengerjaan tugas akhir dan manfaat dari hasil pengerjaan tugas akhir. Selain itu dijelaskan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Sub bab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

1.6.2 Studi Literatur

Dalam pengerjaan Tugas Akhir ini akan dilakukan studi literatur yang dibutuhkan untuk membangun perangkat lunak, yaitu Mininet, *platform* POX, dan *SDN Controller*.

1.6.3 Analisis dan Desain Perangkat Lunak

Dalam pengerjaan tugas akhir ini akan dilakukan analisis dan desain perangkat lunak yang akan dibagun yaitu *SDN Controller* menggunakan *platform* POX. Analisis dan desain perangkat lunak yang akan dilakukan dengan urutan sebagai berikut:

1. Pembangunan *Virtual Network*.
2. Pembangunan *SDN Controller*.
3. Implementasi *SDN Controller* untuk *Virtual Data Center.*

1.6.4 Implementasi Perangkat Lunak

Implementasi Perangkat Lunak merupakan tahapan untuk membangun rancangan program yang telah dibuat. Pada tahapan ini merealisasikan apa yang terdapat pada tahapan sebelumnya,

sehingga menjadi sebuah program yang sesuai dengan apa yang telah direncanakan.

1.6.5 Pengujian dan Evaluasi

Pada tahapan ini dilakukan uji coba pada alat yang telah dibuat. Tahapan ini dimaksudkan untuk mengevaluasi tingkat akurasi dan performa dari alat tersebut serta mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

1.6.6 Penyusunan Buku

Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan tugas akhir yang mencangkup seluruh konsep, dasar teori, implementasi, serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan

Penulisam buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir scara menyeluruh. Selain itu, diharapkan dapat bermanfaat bagi pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan tugas akhir.

2. Bab II. Tinjauan Pustaka
   Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

3. Bab III. Perancangan Perangkat Lunak
   Bab ini berisi implementasi dari perancangan perangkat lunak yang telah dibuat pada bab sebelumnya

4. Bab IV. Implementasi
   Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.

8. Lampiran
   Dalam lampiran terdapat kode sumber program secara keseluruhan.

BAB II
TINJAUAN PUSTAKA

Pada bab ini, dijabarkan tentang penjelasan teori-teori yang berkaitan dengan pokok bahasan tugas akhir. Bab ini juga menjelaskan modul dan alat yang nantinya akan digunakan pada tahap implementasi program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap alat yang digunakan dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Virtualisasi

Virtualisasi adalah sebuah proses membuat representasi berbasis perangkat lunak dari sebuah perangkat keras[4]. Virtualisasi dapat digunakan untuk mem-virtualisasikan aplikasi, server, penyimpanan, maupun jaringan.

Virtulisasi memiliki banyak keuntungan. Virtualisasi dapat menurunkan biaya IT dan dapat meningkatkan efisiensi penggunaan perangkat keras. Virtualisasi juga dapat meningkatkan daya adaptasi dari perangkat lunak.

Jaringan Virtual adalah bagian dari implementasi virtualisasi[5]. Jaringan virtual bertujuan untuk mengganti pengaturan jaringan tidak menggunakan perubahan fisik dalam koneksi ke perangkat keras, namun menggunakan perangkat lunak.

2.2 OpenFlow

OpenFlow adalah sebuah standar yang digunakan untuk mengimplementasikan sebuah protokol jaringan eksperimental[6]. OpenFlow memungkinkan pemisahan antara *control plane*, bagian yang bertugas untuk mengatur fungsi *routing*, dengan *data plane*, bagian yang bertugas untuk mengirim dan meneruskan data. OpenFlow adalah standar pertama yang digunakan untuk implementasi Software-Defined Networking.

2.3 Software-Defined Networking (SDN)

Software-Defined Networking (SDN) adalah sebuah arsitektur jaringan dengan memanfaatkan pemisahan *control plane* dan *data plane*[7]. SDN *Controller* adalah perangkat lunak yang mengatur *control plane* dalam sebuah jaringan, yang memungkinkan seorang *network administrator* untuk mengatur jaringan dari sebuah terminal *SDN Controller*, sehingga meningkatkan fleksibilitas dari jaringan.

2.4 Python

Python adalah sebuah bahasa *scripting* level tinggi[8]. Python memfokuskan pada readability(kemudahan dibaca/dipahami) *script*, dan syntax yang pendek. Python memungkinkan banyak paradigma pemrograman, seperti *object oriented, imperative, procedural,* dan *functional programming*. *Interpreter* Python dapat dijalankan pada banyak sistem operasi, seperti MacOS, Unix, Linux, dan Windows. Keuntungan ini membuat menjadi bahasa yang mudah dipelajari dan diminati banyak orang.

2.5 POX

POX adalah sebuah *platform* pengembangan SDN *Controller* berbasis bahasa Python[2]. POX adalah lanjutan dari NOX yang berbasis C++. POX yang berbasis bahasa python memungkinkan pembangunan dan *prototyping* yang cepat, sehingga menjadi sebuah platform yang diminati oleh pengembang untuk membuat SDN *Controller*.

POX memiliki banyak keuntungan. POX memiliki beberapa komponen yang dapat digunakan ulang sesuai kebutuhan. POX juga telah menyediakan komponen untuk penemuan topologi. POX juga dapat menggunakan metode visualisasi seperti yang digunakan oleh NOX. POX juga dapat berjalan pada semua platform yang dapat menjalankan bahasa Python. Walaupun

dibangun dalam bahasa Python, POX memiliki performa yang tidak jauh berbeda dengan NOX. Semua keuntungan memungkinkan untuk pembuatan perangkat lunak SDN *Controller* yang akan diimplementasikan untuk mengatur jaringan pada sebuah *virtual data center*.

2.6 Mininet

Mininet adalah sebuah platform virtualisasi jaringan[9]. Mininet dapat melakukan virtualisasi dari *host, switch,* dan *link* pada sebuah sistem operasi yang berjalan pada sebuah mesin (Virtual, cloud, maupun mesin nyata). Mininet memungkinkan periset membangun sebuah topologi jaringan virtual dan melakukan pengetesan. Mininet berbasis pada bahasa Python sehingga mudah untuk digunakan dan dikembangkan. Mininet juga sudah dapat memvitrtualisasi *switch* virtual yang mendukung teknologi OpenFlow sehingga dapat digunakan untuk mengembangkan jaringan virtual yang menggunakan metode SDN. Keuntungan ini memungkinkan penggunaan Mininet untuk pembuatan sebuah *virtual data center* yang menggunakan teknologi SDN untuk pengaturan jaringannya.

2.7 Software Container

Software Container adalah sebuah paket yang *lightweight, stand-alone,* dan *executable* dari sebuah perangkat lunak yang sudah memiliki seluruh *dependencies* yang dibutuhkan untuk menjalanakan perangkat lunak tersebut, seperti *web server, library, database,* dan lain lain[10]. Software container memiliki tujuan untuk mengatasi masalah "*perangkat lunak ini berjalan di komputer saya"*.

Software container memiliki banyak keuntungan. Keuntungan pertama adalah meningkatkan fleksibilitas *deployment* dari sebuah perangkat lunak. Software container memastikan perangkat lunak dapat berjalan secara konsisten, tanpa

melihat *environment* container tersebut dijalankan. Software container juga memiliki kelebihan yaitu *lightweight* terutama bila dibandingkan dengan *Virtual Machine*, karena dapat berjalan tanpa perlu memiliki kernel sendiri, sehingga menghemat sumber daya komputasi.

Docker adalah salah satu contoh dari software container yang paling terkenal. Docker adalah platform software container yang dapat berjalan pada Linux maupun Windows. Docker digunakan oleh perusahaan untuk mempercepat proses pengembangan dan *deployment* dari aplikasinya[11].

2.8 Containernet

Containernet[12] adalah sebuah *fork* dari Mininet[13]. *Container* memungkinkan Mininet menggunakan *Software Container* sebagai *host*. Containernet melakukan integrasi dengan cara men-*subclass* kelas *host*. Containernet menambahkan beberapa fungsionalitas dari Mininet yang terkait dengan *software container*, seperti pengaturan jaringan *container*, alokasi sumber daya *container*, penggantian topologi *on-the-fly*, kontrol trafik, dan lain lain.

2.9 OpenWRT

OpenWRT adalah Sistem Operasi untuk perangkat embedded yang berfungsi untuk melakukan *routing* trafik jaringan. OpenWRT menyelesaikan permasalahan yang dialami *developer* terkait konfigurasi perangkat keras yang berbeda – beda untuk tiap perangkat. OpenWRT memungkinkan kustomisasi perangkat untuk melakukan riset ataupun penggunaan biasa[14].

2.10 MySQL

MySQL adalah Sistem Manajemen Database Relasional (RDBMS) yang bersifat *open-source*. MySQL adalah salah satu DBMS paling pupuler di dunia, dengan lebih dari 6 juta pengguna

dari seluruh dunia. MySQL digunakan oleh banyak organisasi besar yang terus berkembang, seperti Facebook, Google, Adobe, dan Zappos[15].

MySQL adalah sebuah DBMS yang dibangun dengan menganut konsep *Structured Query Language*(SQL). Konsep SQL mengutamakan pada pemilihan dan memasukkan data. MySQL memungkinkan pengolahan data dapat dikerjakan secara mudah dan otomatis. MySQL memiliki banyak keuntungan, yaitu portabilitas, skalabilitas, *open-source*, dan lain – lain.

2.11 Hypertext Markup Language(HTML)

HTML adalah bahasa Markup yang digunakan sebagai standar pembuatan aplikasi web[16]. HTML adalah blok pembangun dari setiap aplikasi web. Dengan HTML, gambar dan objek lain, seperti form interaktif, bisa ditampilkan dalam laman web. HTML mendeskripsikan struktur dari web secara semantik, dan pada awalnya ditujukan untuk membuat tampilan dari web.

HTML5 adalah iterasi terbaru dari proses pembaharuan HTML. HTML5 ditujukan untuk menyelesaikan kekacauan yang ada dalam proses pembuatan web, yaitu dengan membuat spesifikasi standar untuk *common practices*.

2.12 Bootstrap

Bootstrap adalah kerangka kerja(*framework*) yang digunakan untuk membuat tampilan *front-end* dari sebuah website[17]. Bootstrap dibangun dengan gabungan dari HTML, CSS, dan JavaScript. Bootrstrap adalah sebuah *framework* yang sangat digemari karena memiliki sifat responsif, yaitu dapat beradaptasi baik ketika ditampilkan di desktop maupun perangkat *mobile*.

2.13 Flask

Flask adalah sebuah kerangka kerja mikro aplikasi web yang dibangun dari Jinja2, Werkzeug, dan dilisensi dengan lisensi BSD[18]. Flask dibangun diatas bahasa pemrograman Python. Flask digunakan pada organisasi besar seperti Pinterest dan LinkedIn.

Flask disebut kerangka kerja mikro karena tidak membutuhkan *library* atau tools lain. Flask tidak memiliki *layer* abstraksi basis data, validasi *form*, maupun komponen lain untuk pembuatan web. Untuk mengatasi kekurangan itu, Flask mampu mengimplementasikan ekstensi maupun *library* lain. Karena dibangun dengan bahasa Python, Flask merupakan kerangka kerja yang mudah dipelajari dan cocok untuk digabungkan dengan aplikasi berbasis Python lainnya.

2.14 Jinja2

Jinja2 adalah bahasa t*emplating* yang modern dan *designer-friendly* untuk Python[19]. Jinja2 dilisensi dengan lisensi BSD. Jinja2 mirip dengan Django namun menggunakan *syntax* yang mirip dengan Python, sehingga mudah dipelajari dan digunakan. Sebagai bahasa *templating*, Jinja2 dapat digunakan untuk membuat *Markup* maupun *Source Code*.

2.15 Redis

Redis adalah media penyimpanan didalam memory yang dapat digunakan untuk basis data, *cache*, maupun pengirim data[20]. Redis merupakan sebuah perangkat lunak *open-source* yang dibangun oleh Redis Labs. Redis mendukung struktur data seperti *list*, *hash*, *string*, *sets*, dan lain lain. Sebagai sebuah pengirim data, Redis dapat digunakan untuk mengirim data dari satu aplikasi ke aplikasi lain, contohnya dari Flask ke Containernet.

2.16 Celery

Celery adalah perangkat lunak *task queue* asinkron yang berdasar pada pengiriman pesan terdistribusi[21]. Celery dibuat menggunakan bahasa Python, namun dapat digunakan untuk beberapa bahasa lain, seperti Ruby, PHP, maupun Node.js. Celery dapat diguakan untuk pengiriman pesan secara sinkron maupun asinkron. Celery difokuskan pada *task* yang bersifat *real-time*, namun juga mendukung penggunaan penjadwalan. Celery digunakan oleh beberapa organisasi besar, seperti Instagram, Mozilla add-ons, dan AdRoll

2.17 MySQLdb *Library*

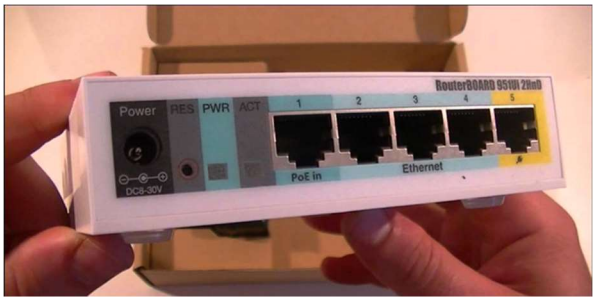
MySQLdb adalah sebuah *Library* untuk bahasa pemrograman Python yang digunakan untuk menghubungkan MySQL dengan bahasa pemrograman Python[22]. MySQLdb tidak tersedia pada instalasi awal Python, sehingga harus dipasang sendiri pasca instalasi. MySQLdb memungkinkan Python untuk mengeksekusi perintah pada MySQL seperti *create, read, update,* dan *delete*.

2.18 MikroTik RouterBOARD 951Ui 2HnD

MikroTik adalah sebuah perusahaan Latvia yang bergerak di bidang *router* dan sistem ISP[23]. MikroTik sekarang menyediakan perangkat keras maupun perangkat lunak untuk konektivitas internet hampir di seluruh dunia. Pada 1997, MikroTik menciptakan RouterOS, sebuah perangkat lunak yang menyediakan stabilitas, fleksibilitas, dan kontrol untuk hampir semua bentuk *interface* dan *routing* data.

MikroTik RouterBOARD adalah perangkat keras yang dibuat oleh MikroTik pada 2002. RouterBOARD seri 951Ui 2HnD memiliki lima *port* ethernet, sebuah *port* USB 2.0, sebuah 2.4GHz 1000mW 802.11b/g/n *wireless AP* dengan antena, CPU 600MHz,

dan memory sebesar 128MB. Gambar 2.1 menunjukkan sebuah RouterBoard 951Ui-2HnD



Gambar 2.1 RouterBoard 951Ui 2HnD

2.19 Iperf

Iperf[24] adalah sebuah kakas yang digunakan untuk peritungan aktif *bandwidth* maksimum yang dapat dicapai oleh sebuah jaringan dengan protokol IP. Iperf melakukan tes dan membuat laporan terkait *bandwidth*, *loss*, dan parameter lainnya. Iperf dapat berjalan pada platform Windows, Linux, MacOS, Android, dan beberapa sistem operasi lain.

BAB III
PERANCANGAN PERANGKAT LUNAK

Bab ini membahas khusus mengenai analisis dan perancangan perangkat lunak yang akan dikembangkan. Secara teknis, aktivitas perancangan merupakan salah satu aktivitas yang sangat penting dilakukan dalam rangka pengembangan perangkat lunak. Beberapa hal yang secara umum dibahas dalam bab ini adalah deskripsi umum sistem, arsitektur umum sistem, diagram kasus penggunaan, perancangan basis data, diagram alur, dan desain antar muka perangkat lunak.

3.1 Deskripsi Umum Sistem

Pada Tugas Akhir ini akan dibangun sebuah perangkat lunak SDN *Controller* yang akan digunakan untuk mengatur jaringan pada sebuah *virtual data center* berbasis *software container*. Tugas Akhir yang akan dibangun dapat dibagi menjadi **4** bagian, yaitu SDN *Controller* yang akan dibangun diatas *platform* POX, web server API untuk pembangunan website yang akan dibangun menggunakan kerangka kerja Flask diatas bahasa pemrograman Python, Implementasi Sistem untuk jaringan *virtual* yang akan dibangun diatas *platform* Containernet, dan implementasi SDN *Controller* untuk switch fisik MikroTik.

SDN *Controller* yang akan dibangun akan memiliki fungsi utama pengaturan jaringan. SDN *Controller* akan mengatur fungsi *routing* pada jaringan. Pada bagian ini akan dijelaskan komponen apa saja yang digunakan POX sehingga dapat dibangun sebuah SDN *Controller* yang cocok untuk lingkungan *virtual data center* berbasis *software container*.

API pembuatan web akan dibangun menggunakan kerangka kerja Flask. API ini berguna untuk menghubungkan jaringan *virtual* yang dibangun dengan tampilan web, dengan cara mengimplementasikan fungsi fungsi pada jaringan *virtual*

sehingga bisa dipanggil menggunakan laman web. API ini juga akan digunakan untuk mengatur fungsi dari SDN Controller yang akan dibangun. Pada API ini juga dibangun basis data untuk standarisasi bentuk data yang akan digunakan pada web dan jaringan virtual.

SDN *Controller* yang dibangun akan diimplementasikan untuk mengatur sebuah jaringan *virtual* yang akan dibangun dengan *platform* Containernet. Pada bagian ini akan dibahas cara konfigurasi yang harus dilakukan agar SDN *Controller* yang dibangun dapat digunakan untuk mengatur fungsi *routing* pada jaringan *virtual.* Pada bagian ini juga akan dijelaskan bagaimana fungsi fungsi yang ada pada API pembuatan web dapat diimplementasikan pada jaringan *virtual*.

SDN *Controller* yang dibangun juga akan diimplementasikan untuk mengatur switch fisik MikroTik. Pada bagian ini akan dibahas kebutuhan, cara konfigurasi, dan langkah langkah yang harus dilakukan agar SDN *Controller* yang dibangun dapat digunakan untuk mengatur fungsi *routing* pada switch fisik.

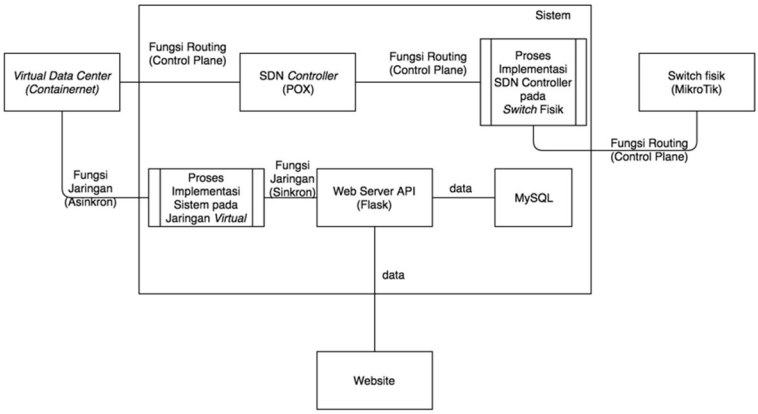
3.2 Arsitektur Umum Sistem

Rancangan arsitektur umum dari sistem dapat dilihat pada Gambar 3.1. Berdasarkan perancangan arsitektur sistem pada Gambar 3.1, Web Server API(Flask) adalah komponen utama dari sistem yang dibangun. Flask akan melakukan komunikasi pertukaran data dengan *web*. Flask juga akan melakukan komunikasi pertukaran data, terutama untuk penyimpanan dan seleksi data, dengan MySQL. Ketika menerima perintah dari web, Flask akan mengirimkan perintah fungsi jaringan, seperti kontrol switch, pada *virtual data center*.

SDN *Controller*(POX), ketika dinyalakan, akan secara otomatis mengatur fungsi *routing* dari virtual data center yang dibangun menggunakan Web Server API(Flask) maupun switch fisik yang terhubung dengan *port* milik SDN *Controller*. SDN

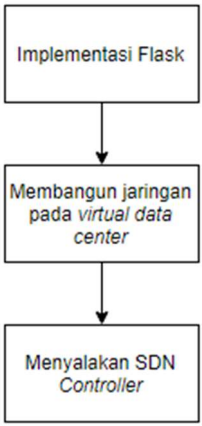
Controller ini akan memiliki konfigurasi yang telah ditentukan sejak awal oleh penulis, sehingga tidak membutuhkan konfigurasi lebih oleh pengguna API.

Terdapat 2 proses untuk Implementasi web server API dan SDN Controller. Implementasi web server API pada *virtual data center* membutuhkan beberapa proses agar dapat mengubah sifat kerja Flask yang sinkron sehingga menjadi asinkron. Implementasi SDN Controller pada *switch* fisik MikroTik membutuhkan proses untuk mengaktifkan fitur OpenFlow pada MikroTik.



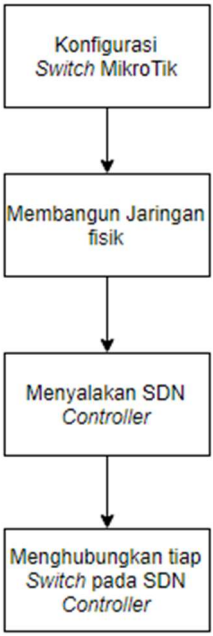
Gambar 3.1 Rancangan Arsitektur Sistem

Terdapat dua cara untuk menggunakan SDN *Controller* yang dibangun, yaitu pada jaringan *virtual* yang dibangun menggunakan Web Server API, dan pada jaringan fisik yang dibangun menggunakan *switch* fisik MikroTik.



Gambar 3.2 Cara penggunaan SDN Controller pada jaringan *virtual*

Gambar 3.2 Menjelaskan penggunaan SDN *Controller* pada jaringa *virtual*. Pengguna pertama tama mengimplementasikan Web Server API yang dibuat menjadi sebuah Website. Lalu membuat *virtual data center* dengan website yang dibangun. Setelah *virtual data* dibangun, dilanjut dengan menyalakan SDN *Controller*. SDN Controller akan secara otomatis terhubung dengan jaringan *virtual data center*.



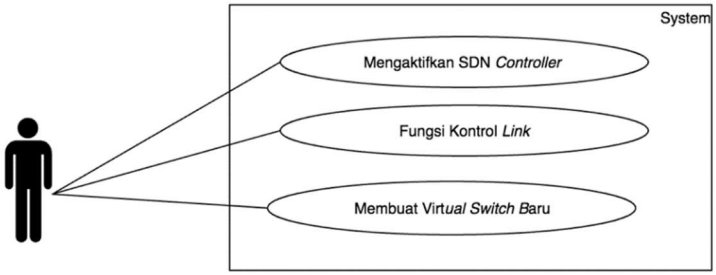
Gambar 3.3 Cara penggunaan SDN Controller pada jaringan fisik

Gambar 3.3 Menjelaskan penggunaan SDN *Controller* pada jaringan fisik yang dibangun menggunakan *switch* fisik MikroTik. Pengguna pertama tama melakukan konfigurasi *switch* fisik MikroTik agar dapat menggunakan protokol OpenFlow yang akan dijelaskan pada bagian 4.5 .Lalu pengguna membangun jaringan dari *switch* MikroTik. Pengguna lalu menyalakan SDN *Controller*. Terakhir, pengguna menyambungkan tiap *switch* MikroTik dengan SDN *Controller* yang dibangun

3.3 Perancangan Web server API (Flask)

Web server API merupakan sebuah API pembuatan website untuk mengatur sistem yang akan dibangun. Web server API akan dibangun menggunakan kerangka kerja Flask. Web server API ini akan memiliki fungsi memantau keadaan serta pengaturan fungsi jaringan dari *virtual data center*. Web server API ini juga akan membuat fungsi jaringan dapat dipanggil di antarmuka website oleh pembuat *virtual data center*.

3.3.1 Perancangan Diagram Kasus Penggunaan (Web server API)



Gambar 3.4 Diagram Kasus Penggunaan

Pada bagian ini akan dijelaskan rincian kasus penggunaan (*use case*) yang akan dibangun dalam web server API. Kasus penggunaan yang akan dibuat dalam website terdiri dari beberapa hal yaitu Membuat *Link* antara *Virtual Switch* dengan *Container*, Membuat *Link* antar *Virtual Switch*, dan membuat *Virtual Switch* baru. Gambar 3.4 merupakan gambar diagram kasus penggunaan (*use case*). Berdasarkan diagram kasus penggunaan yang telah dibuat, berikut ini rincian dari setiap kasus penggunaan yang ada :

3.3.1.1 Mengaktifkan SDN *Controller*

Kasus penggunaan ini menangani fungsi mengaktifkan SDN Controller. SDN Controller yang dimaksud adalah yang ada dalam jaringan *virtual*. SDN Controller ini akan menghubungi SDN Controller yang dibuat menggunakan *platform* POX. Fungsi ini adalah yang mengkaitkan antara SDN Controller yang dibangun dengan jaringan *virtual* yang akan dikembangkan menjadi *virtual data center*.

3.3.1.2 Fungsi Kontrol *Link*

Kasus penggunaan ini menangani kontrol terhadap *link* pada jaringan *virtual*. *Link* yang dimaksud adalah *link* yang menghubungkan dua buah entitas, baik antar *switch* maupun antara *switch* dengan *host* ataupun *container*. Terdapat dua fungsi pada kasus penggunaan ini, yaitu membuat *link* baru, dan menghapus *link* yang ada.

3.3.1.3 Membuat *Virtual* Switch Baru

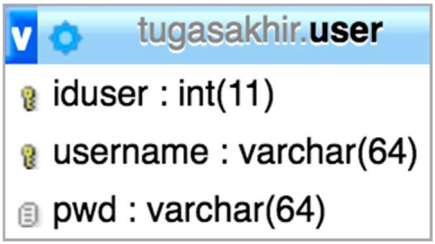
Kasus penggunaan ini menangani fungsi pembuatan *virtual* switch baru pada jaringan *virtual*. Switch ini akan secara langsung mengimplementasikan fungsi SDN dan akan berkomunikasi dengan SDN *Controller* yang dibuat oleh penulis untuk mengatur *control plane* dari switch. Penulis berasumsi bahwa seorang pengguna akan memiliki sebuah *switch*, sehingga pembuatan *switch* akan otomatis dilakukan ketika pengguna mendaftar ke sistem.

3.3.2 Perancangan Basis Data (Web Server API)

Gambar 3.5 Skema Basis Data (Web API)

Perancangan basis data merupakan sebuah langkah yang dilaksanakan demi tercapainya standarisasi data dalam pembuatan web server API. Basis data ini akan digunakan untuk menampung data, sekaligus menseleksi data untuk ditampilkan dalam website. Sistem basis data yang digunakan adalah MySQL. Data yang akan disimpan dalam basis data adalah data pengguna dan *switch* miliknya, , dan data *link* dari *switch* milik pengguna. Gambar 3.5 merupakan rancangan skema dari basis data, dengan penjelasan lebih rinci mengenai setiap entitas dan atribut dalam basis data sebagai berikut ini.

3.3.2.1 Tabel *user*



Gambar 3.6 Skema Tabel *user*

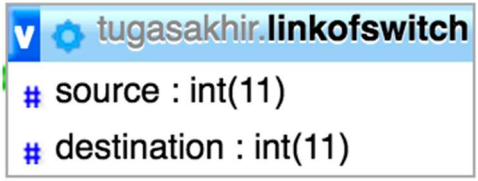
Gambar 3.6 adalah skema dari tabel *user*. Fungsi dari tabel (entity) ini adalah untuk menyimpan data – data dari Pengguna. Data Pengguna yang disimpan di dalamnya meliputi *iduser*,

username, *password*. Tabel 3.1 merupakan tabel yang menjelaskan detail dari tabel *user*.

Tabel 3.1 Detail Tabel *User*

No.	Nama Atribut	Tipe Data	Keterangan
1.	*iduser*	*integer*	*iduser* merupakan *primary key* dari tabel *users*. Diatur sebagai *auto increment*
2.	*username*	*varchar(64)*	*username* dari Pengguna yang digunakan sebagai pengenal untuk masuk (login) ke dalam website
3.	*pwd*	*varchar(64)*	*pwd* adalah *Password* dari Pengguna yang digunakan sebagai kode pengaman untuk masuk (login) ke dalam website

3.3.2.2 Tabel *linkofswitch*



Gambar 3.7 Skema Tabel *linkofswitch*

Gambar 3.7 Skema Tabel *linkofswitch* merupakan skema dari tabel *linkofswitch*. Fungsi dari tabel (entity) ini adalah untuk menyimpan data – data mengenai *link* antar *switch* milik pengguna. Data yang disimpan tersebut meliputi *source* dan *destination*. Data sensor didapatkan saat proses pendaftaran sensor baru. Tabel 3.2 merupakan tabel yang menjelaskan detail dari tabel *sensor*.

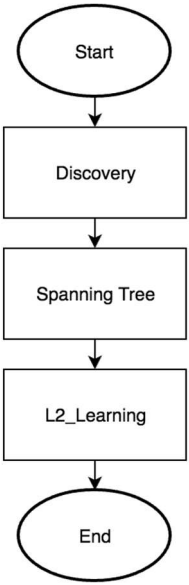
Tabel 3.2 Detail Tabel *linkofswitch*

No.	Nama Atribut	Tipe Data	Keterangan
1	*source*	*integer*	Merupakan salah satu ujung dari *link* antar *switch*. Atribut ini adalah sebuah *foreign key* yang akan berisi *iduser* dari user tersebut.
2.	*destination*	*integer*	Merupakan ujung lain dari *link* antar *switch*. Atribut ini juga akan berisi dari *iduser* dari user tersebut.

3.4 Perancangan SDN Controller

Pada bagian ini akan dilakukan perancangan dari SDN Controller. SDN Controller ini akan dibuat dengan menggunakan *platform* POX beserta komponen komponen milik POX.

SDN Controller adalah inti dari jaringan OpenFlow. SDN Controller akan bertugas untuk melakukan fungsi *routing* pada jaringan. SDN Controller akan dibangun dengan fokus pada lingkungan *virtual data center*.



Gambar 3.8 Diagram Alir Sistem (SDN Controller)

SDN Controller berguna untuk membaca topologi jaringan dan mengatur fungsi routing pada jaringan. SDN Controller dibangun dari komponen yang ada pada POX. Pada Gambar 3.8 Diagram Alir Sistem (SDN Controller), terlihat bahwa terdapat 3 proses atau komponen yang membangun SDN Controller.

Komponen pertama adalah "Discovery". Komponen ini menggunakan protokol LLDP[25] yang dikirim oleh switch untuk menentukan bentuk topologi dari jaringan virtual. Komponen Discovery juga akan mendeteksi apabila terjadi *event link* menyala atau mati. Informasi yang dihasilkan dari komponen ini akan digunakan oleh komponen lain. Komponen ini digunakan karena diperlukan oleh komponen "Spanning Tree" yang akan dijelaskan berikut.

Komponen kedua adalah "Spanning Tree". Komponen ini dibutuhkan ketika topologi jaringan *virtual* memiliki *loop*, sebuah kasus yang sangat mungkin terjadi pada sebuah jaringan besar seperti *virtual data center*. Komponen ini bekerjasama dengan komponen Discovery untuk membentuk *spanning tree* dengan cara menonaktifkan *port* yang tidak ada pada pohon. Opsi *No-Flood* dan *hold-down* digunakan untuk memastikan tidak ada *flood* sebelum terbentuk pohon.

Komponen *Spanning Tree* akan merespon pada perubahan di topologi jaringan. Ketika sebuah *link* rusak, dan jika *link* alternatif tersedia, ia dapat mempertahankan konektifitas pada jaringan dengan membuat pohon baru yang memperbolehkan flooding pada *port* yang terhubung pada *link* alternatif.

Ketika menggunakan komponen *Spanning Tree*, diperlukan komponen *forwarding* yang membuat flow yang memiliki nilai *timeout*. Maka dari itu, kita menggunakan komponen L2_Learning.

Komponen terakhir adalah komponen *forwarding* "L2_Learning". Komponen L2_Learning membuat switch OpenFlow bekerja seperti *Ethernet learning switches*. Komponen ini mempelajari alamat Ethernet MAC, dan mencocokkan semua kolom pada *header* paket sehingga memungkinkan banyak *flow* pada sebuah jaringan untuk setiap pasang alamat MAC. Sebagai contoh, koneksi TCP yang berbeda akan menghasilkan pemasangan *flow* yang berbeda.

Komponen L2_Learning membuat *flow* dengan nilai *timeout* sehingga switch akan secara otomatis menghapus *flow* yang tidak digunakan selama beberapa waktu. Hal ini membuat tabel *flow* lebih mudah dipelihara.

3.5 Perancangan Implementasi Sistem pada Jaringan *Virtual*

Pada bagian ini, akan dijelaskan mengenai rancangan implementasi SDN Controller dan web API pada jaringan *virtual*.

SDN Controller langsung dapat terkoneksi dengan jaringan *virtual* ketika *Controller* pada sebuah jaringan virtual terhubung dengan *port* dari SDN Controller. Sedangkan untuk web API, dibutuhkan penggunaan beberapa perangkat lunak agar dapat terhubung dengan jaringan *virtual*.

Web API yang dibuat diatas kerangka kerja Flask memiliki kinerja yang sinkron. Hal ini akan mengganggu kinerja web karena harus melakukan *pause* proses ketika sedang melakukan *task* pada jaringan virtual. Solusi dalam menghadapi masalah ini adalah dengan menggunakan *task scheduler* bersifat asinkron yaitu Celery[26]. Dengan Celery, Website dapat bekerja terus tanpa perlu menunggu kerja jaringan *virtual* secara sinkron. Dalam mengimplementasikan Celery, dibutuhkan sebuah *message queue*. Dalam Tugas Akhir ini, penulis menggunakan Redis sebagai *message queue*. Gambar 3.9 menggambarkan proses perjalanan *task* dari Web API menuju ke jaringan *virtual*.

Gambar 3.9 Diagram Perancangan Implementasi Sistem Pada Jaringan *Virtual*

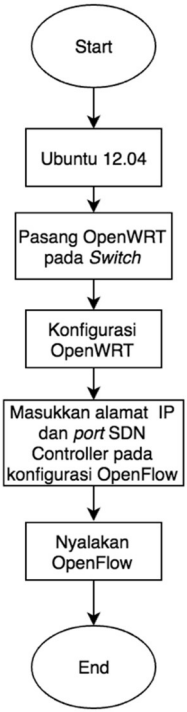
3.6 Perancangan Imlementasi SDN Controller pada Switch Fisik MikroTik

Pada bagian ini, akan dijelaskan mengenai rancangan implementasi SDN Controller pada switch fisik MikroTik. Untuk menjalankan SDN Controller pada switch MikroTik, perlu aktivasi fitur OpenFlow pada switch MikroTik. Aktivasi ini diperoleh menggunakan dua buah cara. Cara pertama adalah dengan melakukan pemasangan perangkat lunak OpenWRT[27] pada switch, lalu melakukan konfigurasi sesuai dengan jenis dari switch

yang digunakan[28]. Cara kedua adalah dengan mengaktifkan fitur OpenFlow yang ada pada RouterOS dengan cara sebagai berikut:

3.6.1 Implementasi SDN Controller pada *switch* Fisik menggunakan OpenWRT

Pada Subbab ini akan dijelaskan mengenai perancangan implementasi SDN Controller pada *switch* fisik Mikrotik dengan OpenWRT. Langkah yang akan dilakukan adalah melakukan pemasangan OpenWRT pada *switch* fisik MikroTik menggunakan Ubuntu 12.04[27]. Lalu akan dilakukan konfigurasi *network* sesuai dengan jenis dari *switch* MikroTik yang digunakan. Lalu memasukkan alamat IP dan *port* dari SDN Controller. Lalu dilanjutkan dengan mengaktifkan fitur OpenFlow pada *switch*. Langkah terakhir adalah menyambungkan dengan SDN Controller yang telah dibuat. Penjelasan per langkah untuk metode ini akan ditulis pada lampiran. Untuk mempermudah penjelasan, dapat memperhatikan diagram alir pada Gambar 3.10 berikut:

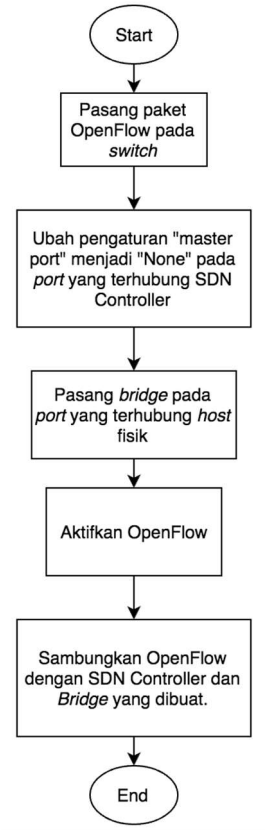


Gambar 3.10 Diagram cara implementasi SDN Controller pada *switch* fisik menggunakan OpenWRT

3.6.2 Implementasi SDN Controller pada *switch* Fisik menggunakan RouterOS

Pada Subbab ini akan dijelaskan mengenai perancangan implementasi SDN Controller pada *switch* fisik MikroTik dengan RouterOS. Langkah yang akan dilakukan adalah pemasangan paket OpenFlow pada *switch*. Lalu mengubah pengaturan *Master Port* pada *port* yang akan dipasang SDN Controller menjadi *None*. Setelah itu, dilakukan pemasangan *bridge* pada semua *port* yag akan dikoneksikan dengan *host* fisik. Lalu mengaktifkan

OpenFlow pada *switch* dan menyambungkan pada *controller* dan *port* yang terhubung dengan *bridge* yang dibuat. Penjelasan detail per langkah akan dilakukan pada bagian lampiran. Untuk mempermudah penjelasan, dapat memperhatikan diagram alir pada Gambar 3.11 berikut:



Gambar 3.11 Diagram cara implementasi SDN Controller pada *switch* fisik menggunakan RouterOS

BAB IV
IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi berupa seluruh langkah yang diperlukan untuk membangun sistem, termasuk *Pseudocode* untuk membangun program, langkah langkah pemasangan perangkat, beserta konfigurasi dari masing masing perangkat.

4.1 Lingkungan Implementasi

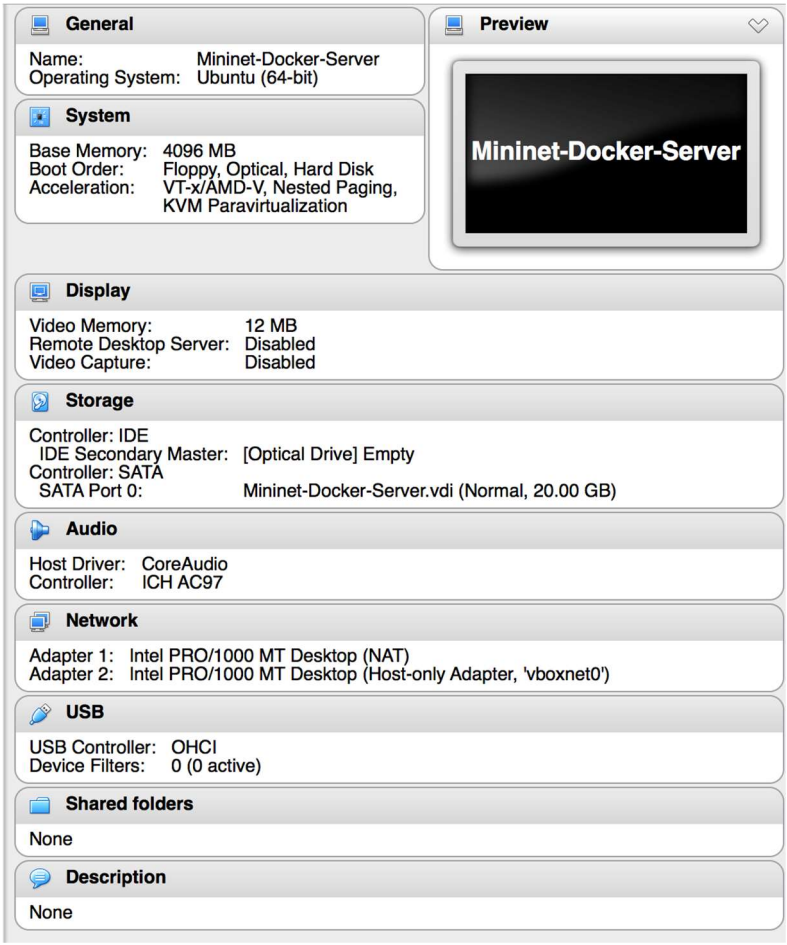
Implementasi POX pada perangkat lunak Software-Defined Networking Controller untuk Data Center berbasis Container menggunakan perangkat keras dan perangkat lunak seperti yang ditunjukkan pada Tabel 4.1:

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i7 @ 2.5 GHz (4 CPUs)
	Memori	16 GB 1600 MHz DDR3
Perangkat Lunak	Sistem Operasi	macOS Sierra v.10.12.14
	Perangkat Pengembang	Oracle VM VirtualBox v. 5.0.28 r111378.

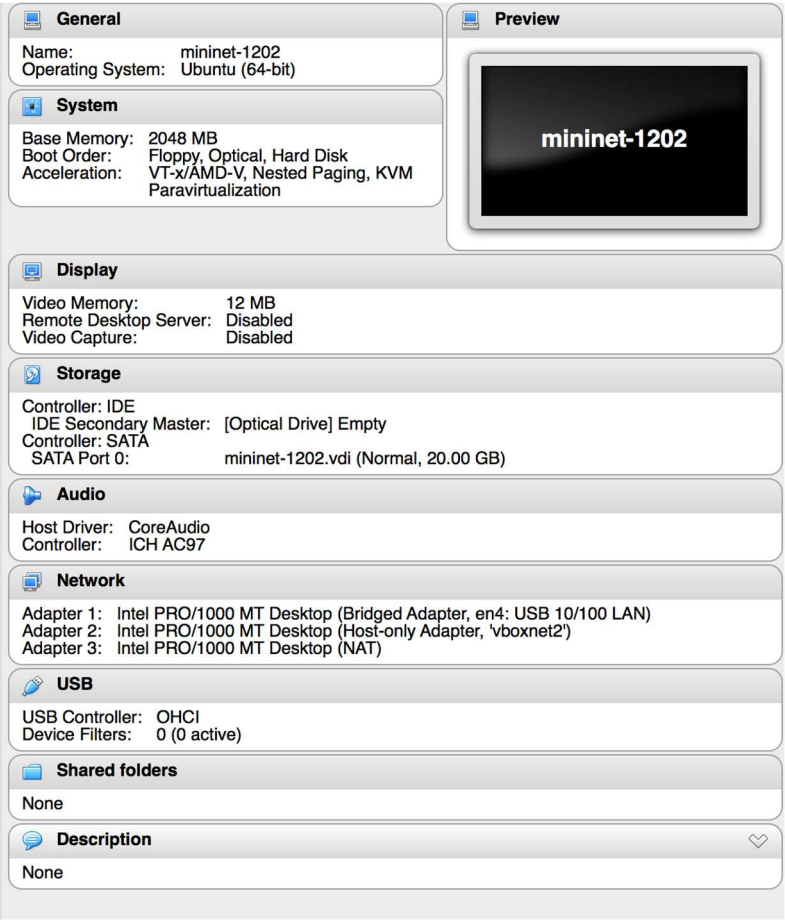
Perangkat keras dan perangkat lunak pada Tabel 4.1 Lingkungan Implementasi Perangkat Lunakakan menjalankan dua buah mesin *virtual* diatas platform virtualisasi VirtualBox. Mesin *virtual* pertama digunakan untuk melakukan pengembangan SDN

Controller dan jaringan *virtual* dengan spesifikasi sebagai pada Gambar 4.1



Gambar 4.1 Spesifikasi Mesin *Virtual* Pengembangan Perangkat Lunak SDN Controller

Mesin *virtual* kedua akan digunakan untuk melakukan pemasangan OpenWRT pada perangkat *switch* fisik MikroTik. Mesin *virtual* ini memiliki spesifikasi sebagai pada Gambar 4.2:



Gambar 4.2 Spesifikasi Mesin *Virtual* Pemasangan OpenWRT pada *Switch* Fisik MikroTik

4.2 Implementasi Web Server API

Pada subbab implementasi Web Server API ini menjelaskan tentang pembangunan perangkat lunak secara detail dan menampilkan *pseudocode* implementasi dari tiap kasus penggunaan pada bagian perancangan dan fungsi yang dibuat untuk menunjang tercapainya implementasi tersebut.

4.2.1 Mengaktifkan Controller

Proses ini menyalakan Controller pada jaringan *virtual* yang nantinya akan dikembangkan menjadi *virtual data center*. Controller ini yang akan menghubungi SDN Controller yang dibangun menggunakan *platform* POX. Tahapannya adalah menerima alamat IP dan *port* dari SDN Controller, dengan menggunakan *RemoteController* sebagai jenis *controller*nya dan TCP sebagai protokolnya. Lalu memanggil Celery untuk membuat sistem menjadi asinkron. Untuk Celery akan dijelaskan lebih lebih lanjut pada bagian 4.4. Langkah terakhir adalah memanggil fungsi *addController* dari Containernet untuk menambah *Controller*. *Pseudocode* dari fungsi ini dapat dilihat pada Pseudocode 4.1:

```
1  FUNCTION startController(self,ipAddress,port):
2      Name <- 'c0'
3      Controller <- RemoteController
4      Protocol <- 'tcp'

5      Celery.desynchronyze()

6      net.addController(
           Name,
           Controller,
           ipAddress,
           protocol,
           port)
7  ENDFUNCTION
```

Pseudocode 4.1 Mengaktifkan Controller

4.2.2 Fungsi Kontrol *Link*

Kasus penggunaan ini terdiri dari dua fungsi, yaitu membuat *link* baru, dan menghapusnya. Penjelasan dari masing – masing fungsi adalah sebagai berikut:

4.2.2.1 Membuat Link Baru

Fungsi ini bertujuan untuk membuat sebuah *link* baru yang menghubungkan dua buah entitas, yaitu antar 2 buah *switch* maupun antara *switch* dan *host*. Langkah pertama dari proses ini adalah dengan memasukkan data *link* ke basis data. Langkah kedua adalah memanggil Celery untuk membuat sistem menjadi asinkron. Langkah terakhir adalah memanggil fungsi *addLink* pada Containernet. *Pseudocode* dari fungsi ini dapat dilihat pada Pseudocode 4.2:

```
1   FUNCTION buatLink(self, source, destination):
2       Database.insert(source,destination)

3       Celery.desynchronyze()

4       net.addLink(source,destination)
5   ENDFUNCTION
```

Pseudocode 4.2 Membuat Link Baru

4.2.2.2 Menghapus *Link* yang ada

Fungsi ini bertujuan untuk menghapus *link* yang sudah ada. Langkah pertama dari proses ini adalah dengan menseleksi data *link* ke basis data lalu menghapusnya. Langkah kedua adalah memanggil Celery untuk membuat sistem menjadi asinkron. Langkah terakhir adalah memanggil fungsi *removeLink* pada Containernet. *Pseudocode* dari fungsi ini dapat dilihat pada Pseudocode 4.3:

```
1   FUNCTION hapusLink(self, source, destination):
2       Database.delete(source,destination)

3       Celery.desynchronyze()

4       net.removeLink(source,destination)
5   ENDFUNCTION
```

Pseudocode 4.3 Menghapus Link

4.2.3 Membuat *Virtual Switch* Baru

Fungsi ini bertujuan untuk membuat *virtual switch* baru. Karena diasumsikan seorang pengguna memiliki sebuah *virtual switch*, maka nama *switch* adalah *id* dari pengguna tersebut. Proses ini dimulai dengan menerima *id* dari user tersebut, lalu menentukan parameter untuk pembuatan *switch*. Lalu memaggil Celery untuk membuat fungsi menjadi asinkron. Lalu memanggil fungsi pembuatan *virtual switch*. Setelah *switch* berhasil dibuat, akan disambungkan dengan *controller* yang dibuat pada bagian 4.2.1 agar dapat terkoneksi dengan SDN Controller dan menerima perintah fungsi *routing* dari SDN Controller. *Pseudocode* dari fungsi ini dapat dilihat pada Pseudocode 4.4:

```
1   FUNCTION buatSwitch(self, userid, controller):
2       switchName <- 's' + userid
3       kelas <- OVSKernelSwitch

4       Celery.desynchronyze()

5       net.addSwitch(switchName, kelas)
6       switchName.start(controller)
7   ENDFUNCTION
```

Pseudocode 4.4 Membuat *Virtual Switch* Baru

4.3 Implementasi SDN Controller

Pada subbab implementasi SDN Controller ini menjelaskan tentang pembangunan SDN Controller yang dibuat dengan *platform* POX. Subbab ini menjelaskan perintah yang harus dijalankan untuk membuat dan menyalakan SDN Controller dari

platform POX dan komponennya. Untuk menyalakan SDN Controller, Pengguna dapat menjalankan perintah seperti pada Kode Sumber 4.1:

```
1   CODE:
2       sudo ~/pox/pox.py \
        forwarding.l2_learning \
        openflow.spanning_tree \
        --no-flood --hold-down \
        openflow.discovery
    END OF CODE
```

Kode Sumber 4.1 Implementasi SDN Controller

4.4 Implementasi Sistem pada Jaringan *Virtual*

Pada Subbab ini akan dijelaskan mengenai langkah langkah yang dibutuhkan untuk mengimplementasikan sistem pada jaringan virtual. Pada Gambar 3.2 terlihat ada 2 bagian yang perlu dijelaskan pada subbab ini, yaitu bagian Redis dan Celery. Untuk Redis dapat langsung dijalankan tanpa perlu konfigurasi lebih lanjut. Sedangkan untuk Celery, dibutuhkan beberapa konfigurasi yang akan dijelaskan.

Penggunaan Celery membutuhkan beberapa persiapan dalam *source code*. Langkah pertama adalah pemilihan *port* untuk komunikasi. Dalam pembuatan sistem ini, digunakan *port* standar Redis. Setelah pemilihan *port* dilakukan, maka akan dilakukan inisialisasi Celery, yang dilanjutkan dengan pembaruan konfigurasi Celery. Setelah semua langkah diatas dilakukan, dilakukan pemanggilan Celery *worker* yang berguna sebagai *task queue* untuk menangani *task* asinkron yang diperintahkan oleh Web Server API. *Pseudocode* dari fungsi ini dapat dilihat pada Pseudocode 4.5 :

```
1   FUNCTION startCelery(self):
2       Celery_Broker <- 'redis://0.0.0.0:6379/0'
3       Celery_Result <- 'redis://0.0.0.0:6379/0'

4       Celery(app.name, app.broker)
5       Celery.config.update()

6       Celery.worker.start()
7   ENDFUNCTION
```

Pseudocode 4.5 Implementasi Sistem pada Jaringan Virtual (Celery)

4.5 Implementasi SDN Controller pada *Switch* Fisik

Pada Subbab ini akan dijelaskan mengenai langkah langkah yang dibutuhkan untuk mengimplementasikan SDN Controller pada *switch* fisik. Terdapat 2 metode agar dapat mengimplementasikan SDN Controller pada perangkat *switch*. Metode pertama adalah dengan melakukan pemasangan OpenWRT pada perangkat *switch*. Metode kedua adalah dengan menggunakan fungsi OpenFlow yang ada pada RouterOS. Subbab ini akan menjelaskan masing masing metode.

4.5.1 Implementasi SDN Controller pada *switch* Fisik menggunakan OpenWRT

Subbab ini menjelaskan cara implementasi SDN Controller pada *switch* fisik MikroTik menggunakan OpenWRT. Untuk langkah implementasi dapat menjalankan langkah langkah yang telah dijelaskan pada bagian 3.6.1 pada bagian perancangan. Untuk konfigurasi yang diperlukan pada OpenWRT dapat melihat pada lampiran bagian 8.5.2

4.5.2 Implementasi SDN Controller pada *switch* Fisik menggunakan RouterOS

Subbab ini menjelaskan cara implementasi SDN Controller pada *switch* fisik MikroTik menggunakan RouterOS. Untuk langkah implementasi dapat menjalankan langkah langkah yang

telah dijelaskan pada bagian 3.6.2 pada bagian perancangan. Untuk konfigurasi yang diperlukan pada RouterOS dapat melihat pada lampiran bagian 8.5.1

(Halaman ini sengaja dikosongkan)

BAB V
HASIL UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai skenario uji coba dan evaluasi pada implementasi SDN Controller pada jaringan *virtual*. Hasil uji coba didapatkan dari implementasi bab 4 dengab skenario pengujian. Bab ini berisikan pembahasan mengenai lingkungan pengujian, skenario pengujian, dan hasil pengujian.

5.1 Lingkungan Pengujian

Lingkungan pengujian mencakup perangkat keras maupun perangkat lunak yang digunakan untuk melakukan uji coba. Uji coba dilakukan pada sebuah mesin virtual yang berjalan diatas perangkat fisik *laptop*, beserta dua buah *switch* MikroTik. Lingkungan uji coba adalah sebagai berikut:

- Laptop MacBook Pro Retina 2015
    - Intel(R) Core(TM) i7 @ 2.5 GHz (4 CPUs)
    - RAM 16 GB 1600 MHz DDR3
    - macOS Sierra v.10.12.14
- Mesin *Virtual* Ubuntu 14.04
    - Ubuntu 14.04 64bit
    - RAM 4 GB
- Dua buah MikroTik RouterBoard 951Ui-2HnD
    - CPU 600 MHz
    - RAM 128 MB
    - 5 Ethernet *port*
    - Menjalankan OpenWRT (1 Buah)
    - Menjalankan RouterOS (1 Buah)

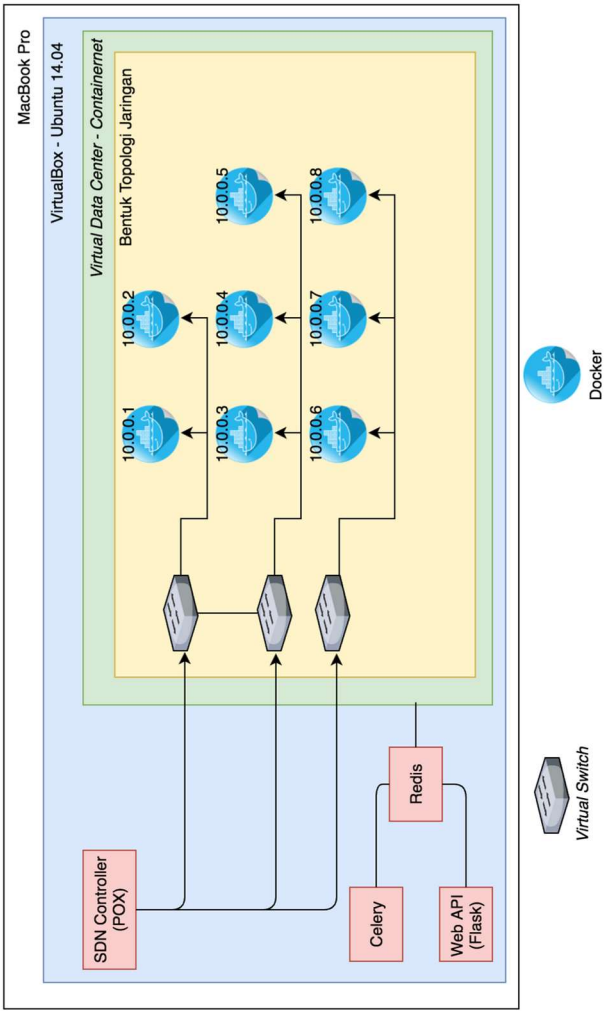
5.2 Arsitektur Pengujian

Arsitektur pengujian mencakup penataan arsitektur tempat pengujian sistem dilaksanakan. Secara garis besar, terdapat dua jenis arsitektur pengujian, yaitu pengujian yang dilakukan untuk

jaringan *virtual*, dan pengujian yang dilakukan untuk *switch* fisik MikroTik. Pengujian pada jaringan *virtual* akan menggunakan mesin *virtual* ubuntu 14.04, sedangkan pengujian *switch* fisik akan dilakukan pada MikroTik yang telah terpasang OpenWRT maupun RouterOS

5.2.1 Arsitektur Pengujian Jaringan *Virtual*

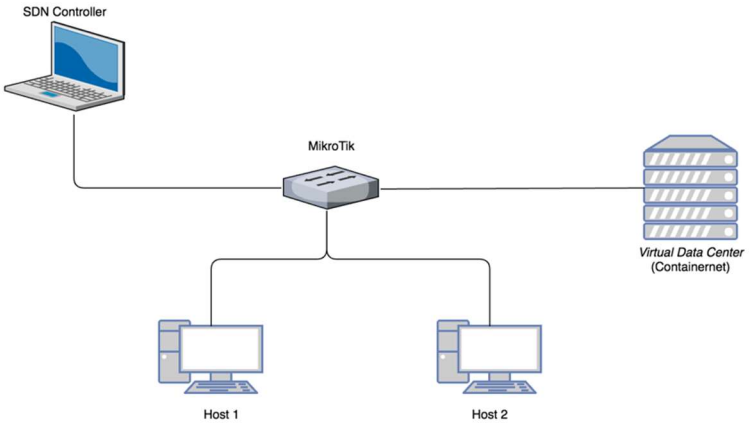
Pengujian jaringan *virtual* akan dijalankan pada sebuah perangkat *MacBook Pro* dengan spesifikasi yang tertera pada subbab diatas. Perangkat *MacBook Pro* akan menjalankan sebuah mesin *virtual* ubuntu yang akan menjalankan Containernet sebagai *Virtual Data Center* dan SDN Controller. Didalam Containernet akan dibentuk topologi jaringan *virtual* yang berubah – ubah sesuai skenario pengujian yang dibangun dari *virtual* switch dan *Software Container*. Untuk mempermudah pemahaman, dapat melihat Gambar 5.1, yang menggambarkan sebuah topologi jaringan garis(*line*) dengan 3 buah *virtual switch*.



Gambar 5.1 Arsitektur Pengujian Jaringan *Virtual*

5.2.2 Arsitektur Pengujian *Switch* Fisik

Pengujian *switch* fisik akan dijalankan menggunakan beberapa perangkat keras. Perangkat pertama adalah *switch* fisik MikroTik yang menggunakan RouterOS maupun OpenWRT sesuai skenario pengujian. Perangkat lain yang ada adalah 3 buah *host*, yaitu 1 buah untuk SDN Controller, dan 2 buah sebagai *host* untuk pengujian. Sebagai Opsi, dapat ditambahkan sebuah perangkat *virtual data center* seperti pada arsitektur diatas. Untuk mempermudah pemahaman, dapat melihat Gambar 5.2 berikut.



Gambar 5.2 Arsitektur Pengujian *Switch* Fisik

5.3 Skenario Pengujian Fungsionalitas

Diperlukan beberapa skenario untuk menguji sistem yang telah dibuat. Pada subbab ini, akan dijelaskan skenario pengujian untuk menguji fungsionalitas dari sistem yang telah dibuat. Hal yang akan diuji adalah keberhasilan sistem menjalankan fungsi kerja sesuai yang dirancang. Skenario pengujian fungsionalitas adalah sebagai berikut:

5.3.1 Skenario Pengujian Fungsionalitas (SF-01) – Tes Koneksi Jaringan Fisik

Skenario pengujian ini menguji kinerja dari implementasi SDN Controller pada jaringan fisik. Hasil yang diharapkan adalah jaringan fisik mampu tersambung dengan SDN Controller dan dapat menerima fungsi routing. Metode yang digunakan untuk uji fungsionalitas ini adalah tes koneksi menggunakan kakas iperf. Tabel 5.1 adalah tabel mengenai skenario uji coba yang akan dilakukan.

Tabel 5.1 Skenario Pengujian Fungsionalitas (SF-01)

ID	**SF-01**
Nama	Uji Coba Implementasi SDN Controller pada jaringan fisik
Tujuan Uji Coba	Menguji fungsionalitas sistem pada jaringan fisik
Kondisi Awal	Terdapat 2 buah *host* fisik yang tersambung pada sebuah *switch* MikroTik yang menggunakan SDN Controller
Metode Pengujian	Uji Koneksi menggunakan Iperf
Lama Pengujian	15 Detik
Skenario	1. Sambungkan *host* pertama ke switch 2. Sambungkan *host* kedua ke switch 3. Buat *Iperf server* pada *host* pertama 4. Lakukan *Iperf client* pada *host* kedua
Masukan	-
Keluaran	Kecepatan rata rata dari jaringan
Hasil yang Diharapkan	Pengujian Iperf berhasil

5.3.2 Skenario Pengujian Fungsionalitas (SF-02) – Tes Koneksi Jaringan *Virtual*

Skenario pengujian ini menguji kinerja dari implementasi SDN Controller pada jaringan *virtual*. Hasil yang diharapkan adalah jaringan *virtual* mampu tersambung dengan SDN Controller dan dapat menerima fungsi routing. Metode yang digunakan untuk uji fungsionalitas ini adalah tes koneksi menggunakan kakas iperf. Tabel 5.2 adalah tabel mengenai skenario uji coba yang akan dilakukan.

Tabel 5.2 Skenario Pengujian Fungsionalitas (SF-02)

ID	**SF-02**
Nama	Uji Coba Implementasi SDN Controller pada jaringan *virtual*
Tujuan Uji Coba	Menguji fungsionalitas sistem pada jaringan *virtual*
Kondisi Awal	Terdapat 2 buah *host virtual* yang tersambung pada dua buah *switch virtual* yang menggunakan SDN Controller
Metode Pengujian	Uji Koneksi menggunakan Iperf
Lama Pengujian	15 Detik
Skenario	1. Sambungkan *host* pertama ke switch pertama 2. Sambungkan *host* kedua ke switch kedua 3. Buat *Iperf server* pada *host* pertama 4. Lakukan *Iperf client* pada *host* kedua
Masukan	-
Keluaran	Kecepatan rata rata dari jaringan
Hasil yang Diharapkan	Pengujian Iperf berhasil

5.3.3 Skenario Pengujian Fungsionalitas (SF-03) – Uji Mengaktifkan SDN Controller

Skenario pengujian ini menguji kinerja dari implementasi SDN Controller. Hasil yang diharapkan adalah SDN Controller mampu mendekteksi jaringan *virtual* yang dibangun. Sistem dinyatakan dapat menjalankan fungsinya apabila SDN Controller yang dibangun dapat mendeteksi jaringan *virtual* yang dibangun. Tabel 5.3 adalah tabel mengenai skenario uji coba yang akan dilakukan.

Tabel 5.3 Skenario Pengujian Fungsionalitas (SF-03)

ID	**SF-03**
Nama	Uji Coba Mengaktifkan SDN Controller
Tujuan Uji Coba	Menguji fungsionalitas SDN Controller yang dibangun
Kondisi Awal	Terdapat sebuah jaringan *virtual* yang belum tersambung dengan SDN Controller
Metode Pengujian	Uji Fungsi "StartController"
Lama Pengujian	-
Skenario	1. Menjalankan fungsi StartController
Masukan	-
Keluaran	-
Hasil yang Diharapkan	SDN Controller mampu mendeteksi jaringan *virtual*

5.3.4 Skenario Pengujian Fungsionalitas (SF-04) – Menambah Switch Virtual

Skenario pengujian ini menguji kinerja dari fungsi penambahan switch. Hasil yang diharapkan adalah switch mampu ditambah dan terhubung pada SDN Controller yang dibangun. Tabel 5.4 adalah tabel mengenai skenario uji coba yang akan dilakukan.

Tabel 5.4 Skenario Pengujian Fungsionalitas (SF-04)

ID	**SF-04**
Nama	Uji Coba Menambahkan *switch virtual*
Tujuan Uji Coba	Menguji fungsionalitas fungsi penambahan switch
Kondisi Awal	Terdapat sebuah jaringan *virtual* yang telah tersambung dengan SDN Controller, Telah masuk ke sistem sebagai seorang *user* yang belum memiliki *switch*
Metode Pengujian	Uji Fungsi "addSwitch"
Lama Pengujian	-
Skenario	1. Menjalankan fungsi addSwitch
Masukan	-
Keluaran	SDN Controller mampu mendeteksi *switch* yang dibuat
Hasil yang Diharapkan	*Switch* berhasil dibuat

5.3.5 Skenario Pengujian Fungsionalitas (SF-05) – Menambah *Link* antar *switch*

Skenario pengujian ini menguji kinerja dari fungsi penambahan *link*. Hasil yang diharapkan adalah *link* mampu ditambah dan terhubung pada SDN Controller yang dibangun. Tabel 5.5 adalah tabel mengenai skenario uji coba yang akan dilakukan.

Tabel 5.5 Skenario Pengujian Fungsionalitas (SF-05)

ID	**SF-05**
Nama	Uji Coba Menambah *link* antar *switch*
Tujuan Uji Coba	Menguji fungsionalitas fungsi menambahkan *link* antar *switch*
Kondisi Awal	Terdapat sebuah jaringan *virtual* yang telah tersambung dengan SDN Controller.
Metode Pengujian	Uji Fungsi "addLink"
Lama Pengujian	-
Skenario	1. Menjalankan fungsi "addLink"
Masukan	-
Keluaran	Sistem menunjukkan bahwa link berhasil dibuat
Hasil yang Diharapkan	*Link* berhasil dibuat

5.3.6 Skenario Pengujian Fungsionalitas (SF-06) – Menghapus *Link* antar *switch*

Skenario pengujian ini menguji kinerja dari fungsi menghapus *link*. Hasil yang diharapkan adalah *link* mampu dihapus dan terdeteksi oleh sistem. Tabel 5.6 adalah tabel mengenai skenario uji coba yang akan dilakukan.

Tabel 5.6 Skenario Pengujian Fungsionalitas (SF-06)

ID	**SF-06**
Nama	Uji Coba Menghapus *link* antar *switch*
Tujuan Uji Coba	Menguji fungsionalitas fungsi menghapus *link* antar *switch*
Kondisi Awal	Terdapat sebuah jaringan *virtual* yang telah tersambung dengan SDN Controller. Tersedia sebuah link yang ingin dihapus
Metode Pengujian	Uji Fungsi "delLink"
Lama Pengujian	-
Skenario	2. Menjalankan fungsi delLink
Masukan	-
Keluaran	Sistem menunjukkan bahwa link berhasil dihapus
Hasil yang Diharapkan	*Link* berhasil dihapus

5.3.7 Skenario Pengujian Fungsionalitas (SF-07) – Uji Kebenaran File Transfer

Skenario pengujian ini menguji kinerja dari jaringan *virtual* yang dibangun. Skenario pengujian ini melakukan pengujian terhadap kebenaran dari file yang dikirim. File akan dikirim dari sebuah Docker ke Docker lainnya pada jaringan virtual menggunakan kakas Netcat[29]. Kesamaan file pada pengirim dan penerima akan diverifikasi menggunakan md5sum[29]. Tabel 5.7 adalah tabel mengenai skenario uji coba yang akan dilakukan.

Tabel 5.7 Skenario Pengujian Fungsionalitas (SF-07)

ID	**SF-07**
Nama	Uji Coba Kebenaran File Transfer
Tujuan Uji Coba	Menguji Kebenaran File Transfer Pada Jaringan Virtual
Kondisi Awal	Terdapat sebuah file pada *host* pertama
Banyak *switch*	1
Metode Pengujian	Uji Kebenaran file transfer Netcat menggunakan md5sum
Lama Pengujian	-
Skenario	1. Sambungkan *host* pertama dan kedua ke switch pertama 2. Buat *netcat server* pada *host* kedua 3. Lakukan pengiriman data via netcat pada *host* kedua 4. Lakukan *hash file* pada *host* pengirim 5. Lakukan *hash file* pada *host* penerima 6. Cocokkan kedua hasil *hash* file
Masukan	-
Keluaran	Hasil *hash file* dari kedua *host*
Hasil yang Diharapkan	Hasil *hash file* dari kedua *host* bernilai sama

5.4 Hasil Pengujian Fungsionalitas

Subbab ini berisi tentang hasil dari pengujian yang telah dilakukan. Pada subbab ini akan berisi hasil dari masing masing skenario pengujian yang dijelaskan pada subbab sebelumnya.

5.4.1 Hasil Pengujian (SF-01) – Tes Koneksi Jaringan Fisik

Sesuai Skenario SF-01, Hasil yang didapatkan dari uji coba yang dilakukan adalah kecepatan dari jaringan yang didapat dari menjalankan kakas iperf pada jaringan fisik. Terlihat bahwa uji iperf berhasil dilakukan, dan dapat disimpulkan bahwa SDN

Controller berhasil diimplementasikan pada jaringan fisik. Hasil pengujian dapat dilihat pada Tabel 5.8:

Tabel 5.8 Hasil Uji Coba SF-01

Kecepatan rata – rata	94.4 Mbps
Status	Berhasil

5.4.2 Hasil Pengujian (SF-02) – Tes Koneksi Jaringan *virtual*

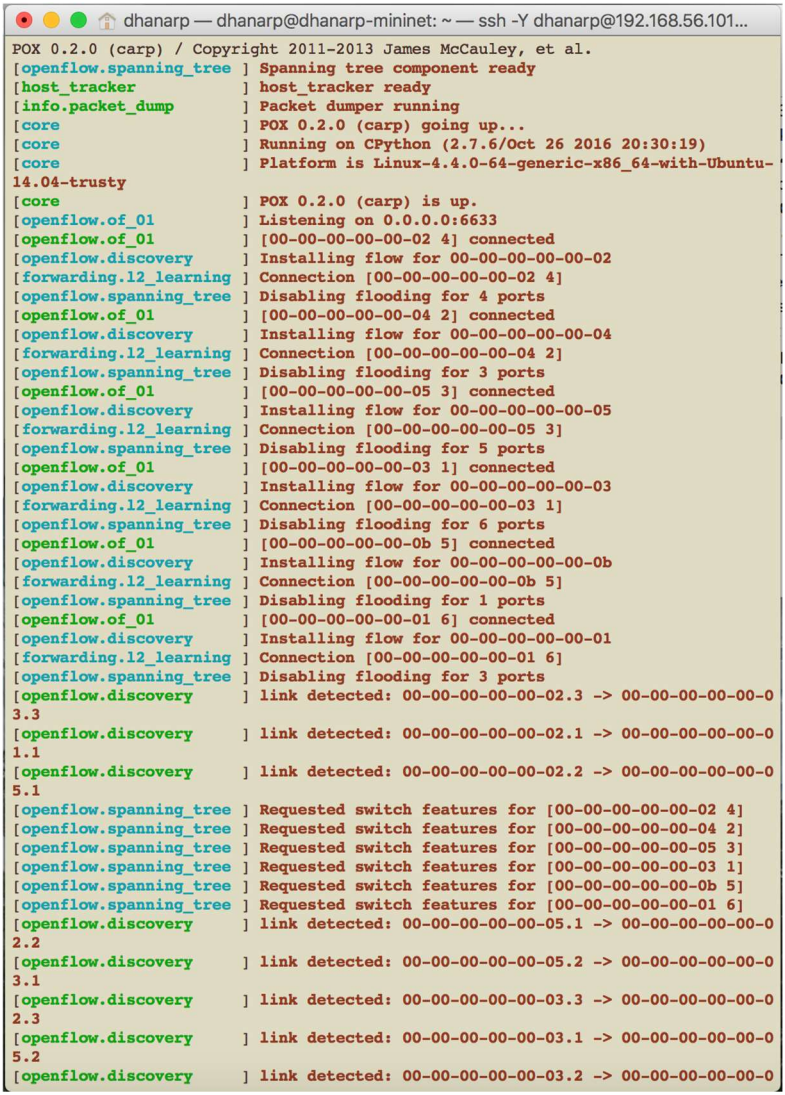
Sesuai Skenario SF-01, Hasil yang didapatkan dari uji coba yang dilakukan adalah kecepatan dari jaringan yang didapat dari menjalankan kakas iperf pada jaringan *virtual*. Terlihat bahwa uji iperf berhasil dilakukan, dan dapat disimpulkan bahwa SDN Controller berhasil diimplementasikan pada jaringan *virtual*. Hasil pengujian dapat dilihat pada Tabel 5.9:

Tabel 5.9 Hasil Uji Coba SF-02

Kecepatan rata – rata	38.0 Gbps
Status	Berhasil

5.4.3 Hasil Pengujian (SF-03) – Uji Mengaktifkan SDN Controller

Sesuai Skenario SF-03, Hasil yang didapatkan dari uji coba yang dilakukan adalah SDN Controller mampu mendeteksi dan tersambung dengan jaringan *virtual*. Gambar 5.3 menunjukkan bahwa SDN Controller berhasil mendeteksi *switch* dan link pada jaringan *virtual* yang dibuat.



Gambar 5.3 Hasil Uji Coba SF-03

5.4.4 Hasil Pengujian (SF-04) – Menambah *Switch Virtual*

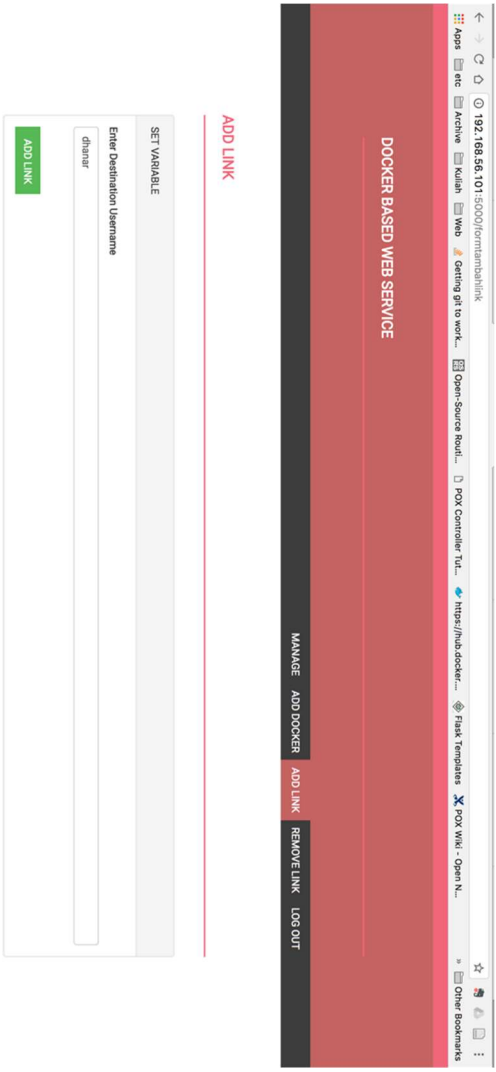
Sesuai Skenario SF-04, Hasil yang didapatkan dari uji coba yang dilakukan adalah SDN Controller mampu mendeteksi dan tersambung dengan *switch* virtual yang ditambahkan. Gambar 5.4 menunjukkan bahwa SDN Controller berhasil mendeteksi *switch* pada jaringan *virtual* yang dibuat.

Gambar 5.4 hasil Uji Menambah *switch virtual*

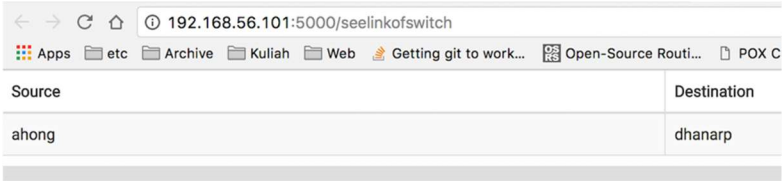
```
[openflow.of_01          ] [00-00-00-00-00-0c 7] connected
[openflow.discovery      ] Installing flow for 00-00-00-00-00-0c
[forwarding.l2_learning  ] Connection [00-00-00-00-00-0c 7]
[openflow.spanning_tree  ] Disabling flooding for 1 ports
[host_tracker            ] Learned 12 65534 3a:49:6a:ed:5f:44
[dump:00-00-00-00-00-0c  ] [ethernet][ipv6][icmpv6][NDNeighborSolicitation]
[dump:00-00-00-00-00-0c  ] [ethernet][ipv6][icmpv6][24 bytes]
[dump:00-00-00-00-00-0c  ] [ethernet][ipv6][icmpv6][24 bytes]
[dump:00-00-00-00-00-0c  ] [ethernet][ipv6][icmpv6][NDRouterSolicitation]
[dump:00-00-00-00-00-0c  ] [ethernet][ipv6][icmpv6][24 bytes]
[openflow.spanning_tree  ] Requested switch features for [00-00-00-00-00-0c 7]
[dump:00-00-00-00-00-0c  ] [ethernet][ipv6][icmpv6][NDRouterSolicitation]
[openflow.spanning_tree  ] Spanning tree updated
[dump:00-00-00-00-00-0c  ] [ethernet][ipv6][icmpv6][NDRouterSolicitation]
```

5.4.5 Hasil Pengujian (SF-05) – Menambah *link* antar *switch*

Sesuai Skenario SF-05, Hasil yang didapatkan dari uji coba yang dilakukan adalah sistem mampu mendeteksi link yang telah dibuat dengan fungsi “addLink”. Gambar 5.5 menunjukkan langkah untuk melakukan pembuatan *link* antara switch user yang telah masuk kedalam sistem dengan switch user tujuan (dhanar). Gambar 5.6 menunjukkan bahwa sistem telah mendeteksi *link* yang dibuat.



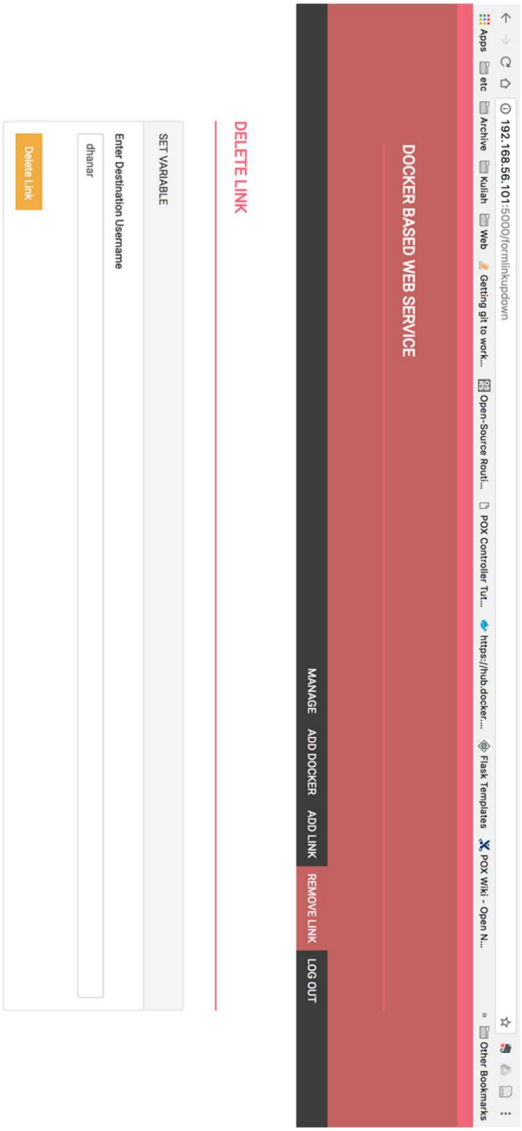
Gambar 5.5 Proses penambahan link



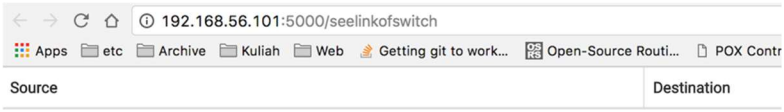
Gambar 5.6 Hasil Uji Coba SF-05

5.4.6 Hasil Pengujian (SF-06) – Menghapus *link* antar *switch*

Sesuai Skenario SF-05, Hasil yang didapatkan dari uji coba yang dilakukan adalah sistem mampu mendeteksi dihapusnya link yang dilakukan dengan fungsi "delLink". Gambar 5.7 menunjukkan langkah untuk melakukan penghapusan *link* antara switch user yang telah masuk kedalam sistem dengan switch user tujuan (dhanar). Gambar 5.8 menunjukkan bahwa sistem telah *link* antara user yang telah masuk (Ahong) dengan switch tujuan (dhanar) telah tiada.



Gambar 5.7 Proses Penghapusan Link



Gambar 5.8 Hasil Uji Coba SF-06

5.4.7 Hasil Pengujian (SP-07) – Uji Keberanan File Transfer

Sesuai Skenario SP-07, didapat hasil *hash file* dari kedua *host* yaitu pengirim dan penerima. Pada Tabel 5.10 bahwa kedua nilai *hash* sama.

Tabel 5.10 Hasil Uji Coba SF-07

Hasil *hash file* pada *host* pengirim	62b651088163e8f21fc2f3187fc3cd7a
Hasil *hash file* pada *host* penerima	62b651088163e8f21fc2f3187fc3cd7a
Status	Sama

5.5 Skenario Pengujian Performa

Diperlukan beberapa skenario untuk menguji sistem yang telah dibuat. Pada subbab ini, akan dijelaskan skenario pengujian untuk menguji kinerja dari sistem yang telah dibuat. Hal yang akan diuji adalah kecepatan dari jaringan dari masing masing skenario pengujian menggunakan kakas Iperf.

5.5.1 Skenario Pengujian Performa (SP-01) – Dua Buah *Switch*

Skenario pengujian ini menguji kinerja dari SDN Controller dan *virtual switch* yang dibangun. Pengujian ini dilakukan untuk menguji kinerja jaringan *virtual* paling kecil yang mungkin, yaitu

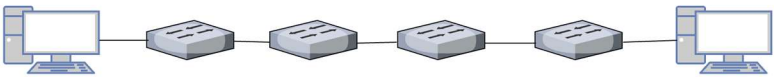
2 buah *switch virtual*. Tabel 5.11 adalah tabel mengenai skenario uji coba yang akan dilakukan.

Tabel 5.11 Skenario Pengujian Performa (SP-01)

ID	**SP-01**
Nama	Uji Coba Jaringan 2 Buah *Switch Virtual*
Tujuan Uji Coba	Menguji kinerja sistem
Kondisi Awal	Terdapat 2 buah *virtual switch* yang terhubung
Banyak *switch*	2
Metode Pengujian	Uji kecepatan menggunakan Iperf
Lama Pengujian	15 Detik
Skenario	3. Sambungkan *host* pertama ke switch pertama 4. Sambungkan *host* kedua ke switch ke 2 5. Buat *Iperf server* pada *host* pertama 6. Lakukan *Iperf client* pada *host* kedua
Masukan	-
Keluaran	Kecepatan rata rata dari jaringan
Hasil yang Diharapkan	-

5.5.2 Skenario Pengujian Performa (SP-02) – Topologi Garis, 20 *Switch*

Skenario pengujian ini menguji kinerja dari SDN Controller dan *virtual switch* yang dibangun dalam bentuk topologi jaringan garis(*line*). Gambar 5.9 merepresentasikan bentuk dari topologi garis. Pada pengujian ini, akan terdapat 20 buah switch dalam topologi garis. Tabel 5.12 adalah tabel mengenai skenario uji coba yang akan dilakukan.



Gambar 5.9 Topologi Garis

Tabel 5.12 Skenario Pengujian Performa (SP-02)

ID	**SP-02**
Nama	Uji Coba Jaringan Topologi Garis – 20 *Switch*
Tujuan Uji Coba	Menguji kinerja sistem terhadap jaringan topologi garis
Kondisi Awal	Terdapat 20 buah *virtual switch* yang terhubung
Banyak *switch*	20
Metode Pengujian	Uji kecepatan menggunakan Iperf
Lama Pengujian	15 Detik
Skenario	1. Sambungkan *host* pertama ke switch pertama 2. Sambungkan *host* kedua ke switch ke 20 3. Buat *Iperf server* pada *host* pertama 4. Lakukan *Iperf client* pada *host* kedua
Masukan	-
Keluaran	Kecepatan rata rata dari jaringan
Hasil yang Diharapkan	-

5.5.3 Skenario Pengujian Performa (SP-03) – Topologi Garis, 100 *Switch*

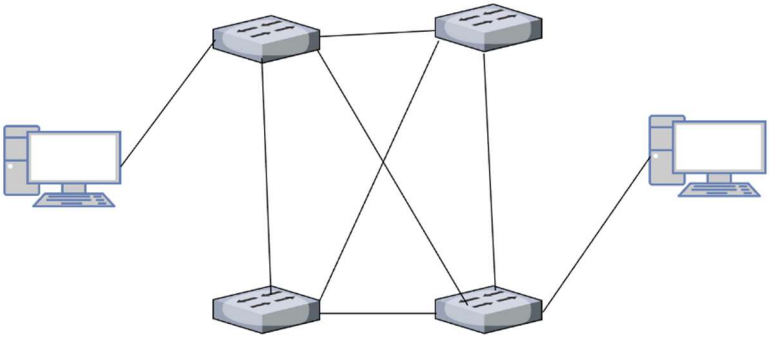
Skenario pengujian ini menguji kinerja dari SDN Controller dan *virtual switch* yang dibangun dalam bentuk topologi jaringan garis(*line*). Gambar 5.9 merepresentasikan bentuk dari topologi garis. Pada pengujian ini, akan terdapat 100 buah switch dalam topologi garis. Tabel 5.13 adalah tabel mengenai skenario uji coba yang akan dilakukan.

Tabel 5.13 Skenario Pengujian Performa (SP-03)

ID	**SP-03**
Nama	Uji Coba Jaringan Topologi Garis – 100 *Switch*
Tujuan Uji Coba	Menguji kinerja sistem terhadap jaringan topologi garis dengan kondisi banyak *switch*
Kondisi Awal	Terdapat 100 buah *virtual switch* dalam topologi garis
Banyak *switch*	100
Metode Pengujian	Uji kecepatan menggunakan Iperf
Lama Pengujian	15 Detik
Skenario	1. Sambungkan *host* pertama ke switch pertama 2. Sambungkan *host* kedua ke switch ke 100 3. Buat *Iperf server* pada *host* pertama 4. Lakukan *Iperf client* pada *host* kedua
Masukan	-
Keluaran	Kecepatan rata rata dari jaringan
Hasil yang Diharapkan	-

5.5.4 Skenario Pengujian (SP-04) – Topologi Jala Tersambung Penuh (*Fully Connected Mesh*)

Skenario pengujian ini menguji kinerja dari SDN Controller dan *virtual switch* yang dibangun dalam bentuk topologi jaringan jala tersambung penuh, yaitu setiap *switch* terhubung dengan semua *switch* lainnya. Gambar 5.10 merepresentasikan bentuk dari topologi jala tersambung penuh. Pada pengujian ini, akan terdapat 20 buah *virtual switch* yang tersambung dalam topologi jala tersambung penuh. Tabel 5.14 adalah tabel mengenai skenario uji coba yang akan dilakukan.



Gambar 5.10 Topologi Jala Tersambung Penuh

Tabel 5.14 Skenario Pengujian Performa (SP-04)

ID	**SP-04**
Nama	Uji Coba Jaringan Jala Tersambung Penuh
Tujuan Uji Coba	Menguji kinerja sistem terhadap jaringan topologi Jala Tersambung Penuh
Kondisi Awal	Terdapat 20 buah *virtual switch* dalam topologi Jala Tersambung penuh
Banyak *switch*	20
Metode Pengujian	Uji kecepatan menggunakan Iperf
Lama Pengujian	15 Detik
Skenario	1. Sambungkan *host* pertama ke switch pertama 2. Sambungkan *host* kedua ke switch ke 20 3. Buat *Iperf server* pada *host* pertama 4. Lakukan *Iperf client* pada *host* kedua
Masukan	-
Keluaran	Kecepatan rata rata dari jaringan
Hasil yang Diharapkan	SDN Controller dapat mengatasi jaringan yang memiliki *looping*.

5.5.5 Skenario Pengujian Performa (SP-05) – Switch Fisik RouterOS

Skenario pengujian ini menguji kinerja dari SDN Controller pada *switch* fisik Mikrotik yang menggunakan RouterOS. Pada pengujian ini, digunakan 1 buah *switch* fisik MikroTik. Tabel 5.15 adalah tabel mengenai skenario uji coba yang akan dilakukan.

Tabel 5.15 Skenario Pengujian Performa (SP-05)

ID	**SP-05**
Nama	Uji Coba Sistem Terhadap *Switch* Fisik RouterOS
Tujuan Uji Coba	Menguji kinerja sistem terhadap *switch* fisik RouterOS
Kondisi Awal	Terdapat 1 buah *switch* fisik RouterOS yang telah terhubung dengan SDN Controller
Banyak *switch*	1
Metode Pengujian	Uji kecepatan menggunakan Iperf
Lama Pengujian	15 Detik
Skenario	1. Sambungkan *host* pertama dan kedua ke switch pertama 2. Buat *Iperf server* pada *host* pertama 3. Lakukan *Iperf client* pada *host* kedua
Masukan	-
Keluaran	Kecepatan rata rata dari jaringan
Hasil yang Diharapkan	-

5.5.6 Skenario Pengujian Performa (SP-06) – Switch Fisik OpenWRT

Skenario pengujian ini menguji kinerja dari SDN Controller pada *switch* fisik Mikrotik yang menggunakan RouterOS. Pada pengujian ini, digunakan 1 buah *switch* fisik MikroTik. Tabel 5.16 adalah tabel mengenai skenario uji coba yang akan dilakukan.

Tabel 5.16 Skenario Pengujian Performa (SP-06)

ID	**SP-06**
Nama	Uji Coba Sistem Terhadap *Switch* Fisik OpenWRT
Tujuan Uji Coba	Menguji kinerja sistem terhadap *switch* fisik OpenWRT
Kondisi Awal	Terdapat 1 buah *switch* fisik OpenWRT yang telah terhubung dengan SDN Controller
Banyak *switch*	1
Metode Pengujian	Uji kecepatan menggunakan Iperf
Lama Pengujian	15 Detik
Skenario	1. Sambungkan *host* pertama dan kedua ke switch pertama 2. Buat *Iperf server* pada *host* pertama 3. Lakukan *Iperf client* pada *host* kedua
Masukan	-
Keluaran	Kecepatan rata rata dari jaringan
Hasil yang Diharapkan	-

5.6 Hasil Pengujian Performa

Subbab ini berisi tentang hasil dari pengujian yang telah dilakukan. Pada subbab ini akan berisi hasil dari masing masing skenario pengujian yang dijelaskan pada subbab sebelumnya.

5.6.1 Hasil Pengujian (SP-01) – Dua Buah *Switch*

Sesuai Skenario SP-01, Hasil yang didapatkan dari uji coba yang dilakukan adalah kecepatan dari jaringan *virtual* yang memiliki 2 buah *switch*. Hasil pengujian dapat dilihat pada Tabel 5.17:

Tabel 5.17 Hasil Uji Coba SP-01

Detik ke -	Kecepatan (Gbps)
0 – 1	38.4
1 – 2	38.6
2 – 3	38.6
3 – 4	37.6
4 – 5	37.5
5 – 6	37.7
6 – 7	38.0
7 – 8	38.6
8 – 9	37.9
9 – 10	38.2
10 – 11	37.8
11 – 12	37.6
12 – 13	37.9
13 – 14	38.5
14 – 15	37.6
Rata Rata	38.0

5.6.2 Hasil Pengujian (SP-02) – Topologi Garis, 20 *Switch*

Sesuai Skenario SP-02, Hasil yang didapatkan dari uji coba yang dilakukan adalah kecepatan dari jaringan *virtual* yang memiliki 20 buah *switch* dalam topologi garis. Terlihat terjadi penurunan pada kecepatan jaringan. Hasil pengujian dapat dilihat pada Tabel 5.18:

Tabel 5.18 Hasil Uji Coba SP-02

Detik ke -	Kecepatan (Gbps)
0 – 1	13.5
1 – 2	21.8
2 – 3	22.0
3 – 4	21.7
4 – 5	21.4
5 – 6	18.8
6 – 7	16.9
7 – 8	19.2
8 – 9	19.0
9 – 10	18.3
10 – 11	18.4
11 – 12	19.5
12 – 13	20.2
13 – 14	20.6
14 – 15	19.3
Rata Rata	19.4

5.6.3 Hasil Pengujian (SP-03) – Topologi Garis, 100 *Switch*

Sesuai Skenario SP-03, Hasil yang didapatkan dari uji coba yang dilakukan adalah kecepatan dari jaringan *virtual* yang memiliki 100 buah *switch* dalam topologi garis. Terlihat terjadi penurunan kecepatan yang cukup signifikan. Hasil pengujian dapat dilihat pada Tabel 5.19:

Tabel 5.19 Hasil Uji Coba SP-03

Detik ke -	Kecepatan (Gbps)
0 – 1	3.56
1 – 2	7.20
2 – 3	6.95
3 – 4	8.27
4 – 5	7.95
5 – 6	7.74
6 – 7	7.37
7 – 8	6.75
8 – 9	7.60
9 – 10	7.66
10 – 11	7.72
11 – 12	7.49
12 – 13	6.79
13 – 14	7.16
14 – 15	7.60
Rata Rata	7.19

5.6.4 Hasil Pengujian (SP-04) – Topologi Jala Terhubung Penuh

Sesuai Skenario SP-04, Hasil yang didapatkan dari uji coba yang dilakukan adalah kecepatan dari jaringan *virtual* yang memiliki 20 buah *switch* dalam topologi jala terhubung penuh. Pengujian menunjukkan bahwa SDN Controller yang dibangun mampu mengatasi jaringan yang memiliki *looping*. Terlihat terjadi penurunan kecepatan yang tidak signifikan ketika dibandingkan dengan jaringan yang memiliki 2 buah *switch*. Hasil pengujian dapat dilihat pada Tabel 5.20:

Tabel 5.20 Hasil Uji Coba SP-04

Detik ke -	Kecepatan (Gbps)
0 – 1	28.5
1 – 2	34.7
2 – 3	34.9
3 – 4	35.0
4 – 5	35.5
5 – 6	32.6
6 – 7	35.8
7 – 8	35.7
8 – 9	35.5
9 – 10	34.5
10 – 11	31.3
11 – 12	34.1
12 – 13	34.8
13 – 14	35.3
14 – 15	34.6
Rata Rata	34.2

5.6.5 Hasil Pengujian (SP-05) – *Switch* Fisik RouterOS

Sesuai Skenario SP-05, Hasil yang didapatkan dari uji coba yang dilakukan adalah kecepatan dari sebuah *switch* fisik Mikrotik dengan RouterOS. Terjadi penurunan kecepatan yang signifikan bila dibandingkan dengan jaringan *virtual*. Hasil pengujian dapat dilihat pada Tabel 5.21:

Tabel 5.21 Hasil Uji Coba SP-05

Detik ke -	Kecepatan (Mbps)
0 – 1	91.1
1 – 2	94.5
2 – 3	95.4
3 – 4	94.2
4 – 5	95.5
5 – 6	94.4
6 – 7	95.4
7 – 8	94.3
8 – 9	95.5
9 – 10	94.4
10 – 11	95.4
11 – 12	94.3
12 – 13	95.4
13 – 14	94.5
14 – 15	95.4
Rata Rata	94.6

5.6.6 Hasil Pengujian (SP-06) – *Switch* Fisik OpenWRT

Sesuai Skenario SP-06, Hasil yang didapatkan dari uji coba yang dilakukan adalah kecepatan dari sebuah *switch* fisik Mikrotik dengan OpenWRT. Terjadi penurunan kecepatan yang signifikan bila dibandingkan dengan jaringan *virtual*. Hasil pengujian dapat dilihat pada Tabel 5.22:

Tabel 5.22 Hasil Uji Coba SP-06

Detik ke -	Kecepatan (Mbps)
0 – 1	88.0
1 – 2	94.4
2 – 3	95.4
3 – 4	94.4
4 – 5	95.4
5 – 6	94.5
6 – 7	95.3
7 – 8	94.4
8 – 9	95.5
9 – 10	94.4
10 – 11	95.4
11 – 12	94.4
12 – 13	95.4
13 – 14	94.4
14 – 15	95.3
Rata Rata	94.4

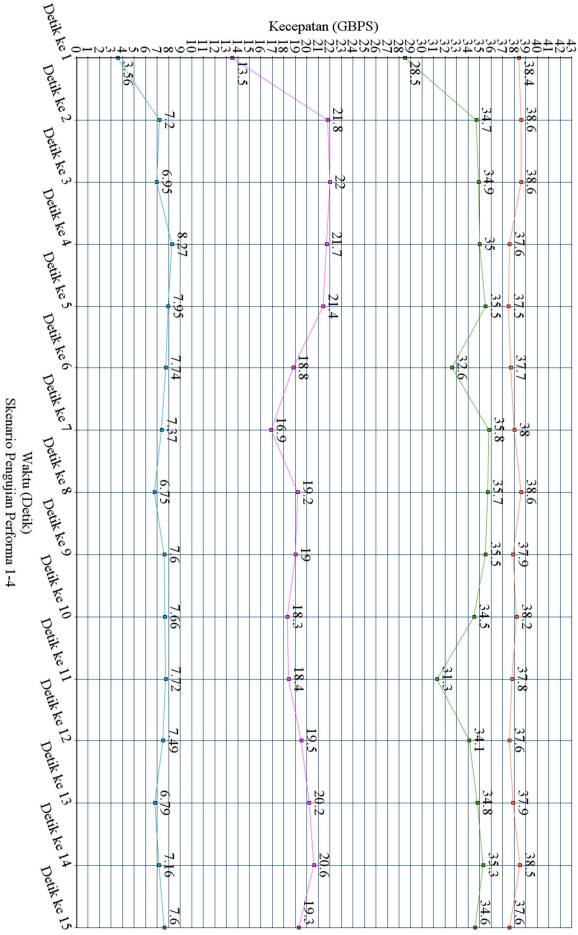
5.7 Kesimpulan Pengujian

Berdasarkan hasil pengujian yang dilakukan dapat ditarik beberapa kesimpulan. Gambar 5.11, Gambar 5.12, dan Gambar 5.13 merupakan graf yang menunjukkan perbandingan hasil masing masing uji coba performa. Berdasarkan hasil pengujian performa, penulis menganalisis beberapa kesimpulan. Kesimpulan yang dapat ditarik adalah sebagai berikut:

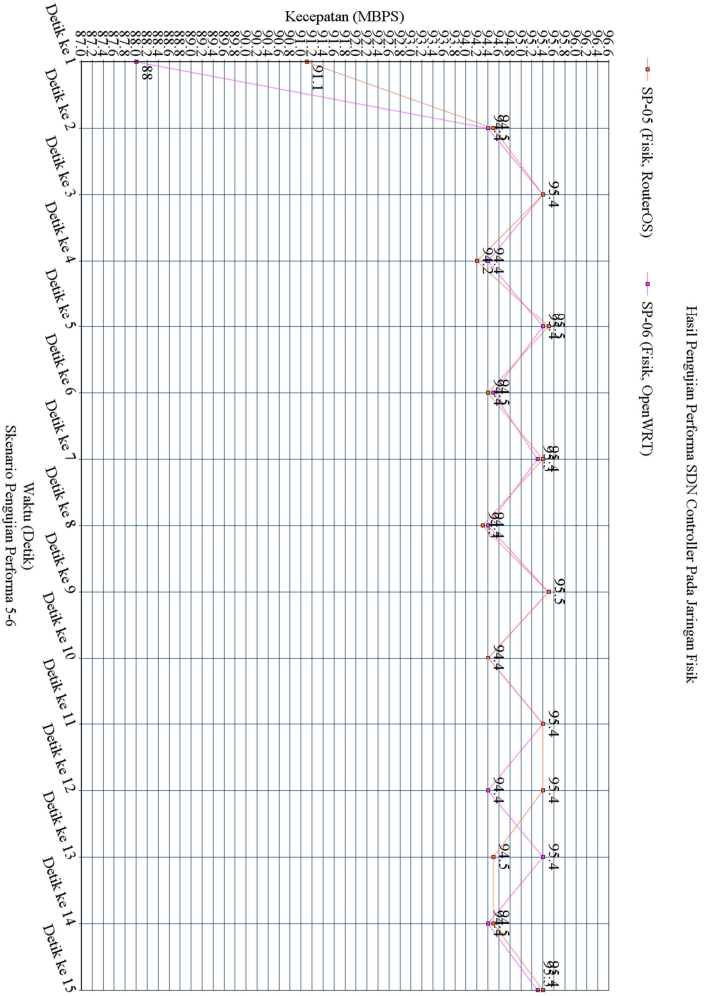
1. Sistem yang dibangun telah berjalan sesuai dengan rancangan. Hal ini dibuktikan oleh Skenario Pengujian Fungsionalitas SF-01 sampai dengan SF-07
2. Jaringan *virtual* memiliki kecepatan jaringan yang jauh lebih tinggi dibanding jaringan fisik. Pada jaringan *virtual*, kecepatan jaringan dapat mencapai 40Gbps, sedangkan pada jaringan fisik, kecepatan maksimum jaringan hanya

100 Mbps. Hal ini ditunjukkan dengan melihat hasil percobaan dari SP-01 sampai dengan SP-06.

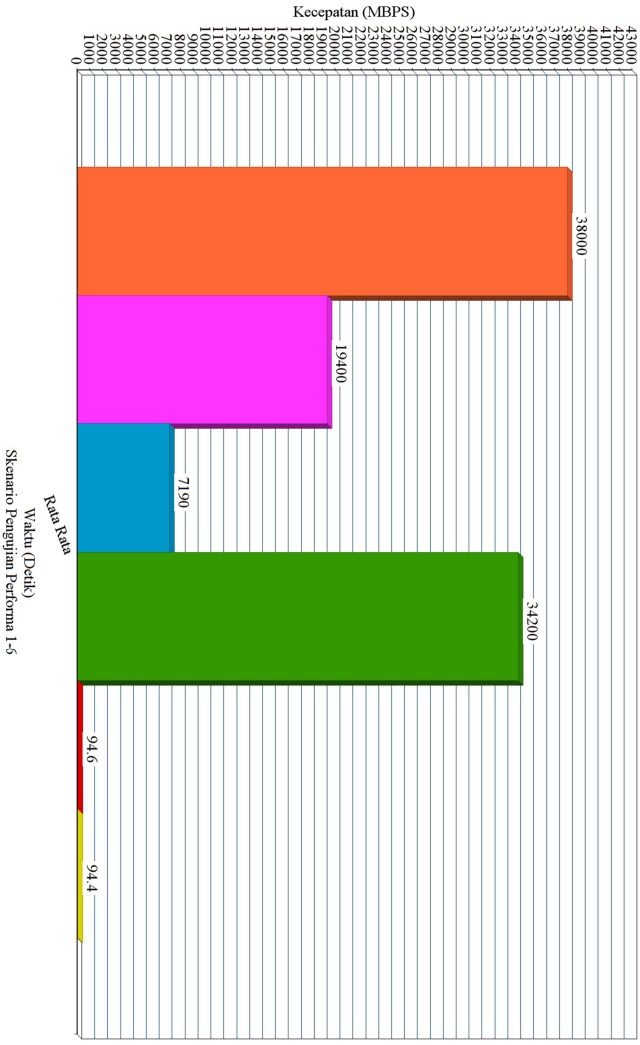
3. Kecepatan jaringan dipengaruhi oleh banyaknya *switch* yang harus dilewati dari pengirim ke penerima (*hops*). Hal ini ditunjukkan oleh percobaan SP-02 dengan SP-04 yang sama sama memiliki 20 *switch* namun memiliki jumlah *hops* yang berbeda. Begitu pula dengan membandingkan percobaan SP-01, SP-02, SP-03, yang sama sama memiliki topologi garis namun memiliki perbedaan jumlah *hops* yaitu 1, 19, dan 99 buah *hops.*
4. Perbedaan bentuk topologi pada jaringan virtual akan sangat mempengaruhi kecepatan jaringan. Topologi yang antar *switch* saling terhubung akan memakan sumber daya komputasi yang besar, namun akan memiliki kecepatan jaringan yang lebih tinggi. Hal ini ditunjukkan oleh SP-02 dan SP-04 yang memiliki jumlah *switch* sama, namun memiliki bentuk topologi berbeda sehingga memiliki jumlah *hops* antar *host* yang berbeda.
5. Jaringan *virtual* yang dibangun dinilai dapat diandalkan, dilihat dari tidak adanya kesalahan pada pengiriman data, yaitu dilihat dari kesamaan hasil *hasil* hash file pada pengirim dan penerima. Hal ini ditunjukkan dengan hasil pengujian SF-07



Gambar 5.11 Hasil Pengujian Performa Sistem Virtual



Gambar 5.12 Hasil Pengujian Performa Sistem Fisik



Gambar 5.13 Hasil Pengujian Performa Sistem

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan. Selain kesimpulan, terdapat juga saran yang ditujukan untuk pengembangan perangkat lunak nantinya.

6.1 Kesimpulan

Kesimpulan yang didapatkan berdasarkan hasil uji coba implementasi POX pada perangkat lunak *Software-Defined Networking* Controller untuk data center berbasis container adalah :

1. *Platform* Containernet dapat digunakan untuk membagun sebuah jaringan *virtual* yang menggunakan arsitektur Software-Defined Networking, yang dapat diimplementasikan untuk membangun *virtual data center* berbasis Container.
2. *Platform* POX dapat digunakan untuk membuat SDN Controller untuk sebuah jaringan *virtual* yang dibangun menggunakan *platform* Containernet.
3. SDN Controller ini sudah dapat mengontrol jaringan *virtual* maupun *fisik*,termasuk jaringan yang memiliki *loop*.
4. Besarnya kecepatan pada jaringan *virtual* yang dibangun ditentukan oleh banyaknya *hops* yang dilewati dari pengirim ke penerima. Hal ini disimpulkan dari melihat hasil pengujian SP-02 dengan SP-04 yang memiliki jumlah *switch* sama namun topologi berbeda sehingga jumah *hops* yang berbeda.

6.2 Saran

Saran yang diberikan terkait pengembangan pada Tugas Akhir ini adalah penulis menyarankan pembuatan *link* antar *switch*

virtual yang sering berkomunikasi data, agar dapat mengurangi jumlah *hops* sehingga meningkatkan kecepatan jaringan.

DAFTAR PUSTAKA

[1] "Software-Defined Networking (SDN) Definition - Open Networking Foundation." [Daring]. Tersedia pada: https://www.opennetworking.org/sdn-resources/sdn-definition. [Diakses: 14-Des-2016].

[2] "What is POX? - Definition from WhatIs.com," *SearchSDN*. [Daring]. Tersedia pada: http://searchsdn.techtarget.com/definition/POX. [Diakses: 02-Mei-2017].

[3] "Mininet Overview - Mininet," *Mininet*. [Daring]. Tersedia pada: http://mininet.org/overview/. [Diakses: 14-Des-2016].

[4] "Virtualization Technology & Virtual Machine Software." [Daring]. Tersedia pada: http://www.vmware.com/solutions/virtualization.html. [Diakses: 02-Mei-2017].

[5] "What Is a Virtual Network?," *SDxCentral*, 10-Mei-2016. [Daring]. Tersedia pada: https://www.sdxcentral.com/sdn/network-virtualization/definitions/what-is-a-virtual-network/. [Diakses: 02-Mei-2017].

[6] "OpenFlow » What is OpenFlow?" .

[7] "SDN articles, whitepapers, videos & more," *SDxCentral*, 21-Feb-2014. [Daring]. Tersedia pada: https://www.sdxcentral.com/sdn/. [Diakses: 02-Mei-2017].

[8] "Welcome to Python.org," *Python.org*. [Daring]. Tersedia pada: https://www.python.org/. [Diakses: 07-Apr-2017].

[9] "Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet." [Daring]. Tersedia pada: http://mininet.org/. [Diakses: 02-Mei-2017].

[10] "What is a Container," *Docker*, 29-Jan-2017. [Daring]. Tersedia pada: https://www.docker.com/what-container. [Diakses: 02-Mei-2017].

[11] "What is Docker?," *Docker*, 14-Mei-2015. [Daring]. Tersedia pada: https://www.docker.com/what-docker. [Diakses: 14-Des-2016].
[12] M. Peuster, H. Karl, dan S. van Rossem, "MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2016, hal. 148–153.
[13] "containernet/containernet," *GitHub*. [Daring]. Tersedia pada: https://github.com/containernet/containernet. [Diakses: 03-Mei-2017].
[14] "OpenWrt." [Daring]. Tersedia pada: https://openwrt.org/. [Diakses: 03-Mei-2017].
[15] "MySQL :: Why MySQL?" [Daring]. Tersedia pada: https://www.mysql.com/why-mysql/. [Diakses: 03-Mei-2017].
[16] "W3C HTML." [Daring]. Tersedia pada: https://www.w3.org/html/. [Diakses: 03-Mei-2017].
[17] "Bootstrap · The world's most popular mobile-first and responsive front-end framework." [Daring]. Tersedia pada: http://getbootstrap.com/. [Diakses: 03-Mei-2017].
[18] "Welcome | Flask (A Python Microframework)." [Daring]. Tersedia pada: http://flask.pocoo.org/. [Diakses: 03-Mei-2017].
[19] "Welcome to Jinja2 — Jinja2 Documentation (2.9)." [Daring]. Tersedia pada: http://jinja.pocoo.org/docs/2.9/. [Diakses: 03-Mei-2017].
[20] "Redis." [Daring]. Tersedia pada: https://redis.io/. [Diakses: 04-Mei-2017].
[21] "Homepage | Celery: Distributed Task Queue." [Daring]. Tersedia pada: http://www.celeryproject.org/. [Diakses: 04-Mei-2017].
[22] A. Dustman, "MySQLdb: a Python interface for MySQL," *MySQLdb User's Guide*. [Daring]. Tersedia pada:

http://mysql-python.sourceforge.net/MySQLdb.html. [Diakses: 08-Apr-2017].
[23] "MikroTik." [Daring]. Tersedia pada: http://www.mikrotik.com/. [Diakses: 04-Mei-2017].
[24] "iPerf - The TCP, UDP and SCTP network bandwidth measurement tool." [Daring]. Tersedia pada: https://iperf.fr/. [Diakses: 30-Mei-2017].
[25] "LinkLayerDiscoveryProtocol - The Wireshark Wiki." [Daring]. Tersedia pada: https://wiki.wireshark.org/LinkLayerDiscoveryProtocol. [Diakses: 15-Mei-2017].
[26] "Using Celery With Flask - miguelgrinberg.com." [Daring]. Tersedia pada: https://blog.miguelgrinberg.com/post/using-celery-with-flask. [Diakses: 17-Mei-2017].
[27] "SDN.pdf." .
[28] "Table of Hardware [OpenWrt Wiki]." [Daring]. Tersedia pada: https://wiki.openwrt.org/toh/start. [Diakses: 17-Mei-2017].
[29] "Netcat: the TCP/IP swiss army." [Daring]. Tersedia pada: http://nc110.sourceforge.net/. [Diakses: 31-Mei-2017].
[30] "HowToMD5SUM - Community Help Wiki." [Daring]. Tersedia pada: https://help.ubuntu.com/community/HowToMD5SUM. [Diakses: 31-Mei-2017].

(Halaman ini sengaja dikosongkan)

LAMPIRAN

8.1 HTML WEB UI

Kode Sumber 8.1 login-gui.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1, maximum-scale=1" />
    <meta name="description" content="" />
    <meta name="author" content="" />
    <!--[if IE]>
        <meta http-equiv="X-UA-Compatible"
content="IE=edge,chrome=1">
        <![endif]-->
    <title>Docker Based Mininet with SDN
Controller</title>
    <!-- BOOTSTRAP CORE STYLE  -->
    <link href="{{ url_for('static',
filename='css/bootstrap.css') }}" rel="stylesheet" />
    <!-- FONT AWESOME ICONS  -->
    <link href="{{ url_for('static', filename='css/font-
awesome.css') }}" rel="stylesheet" />
    <!-- CUSTOM STYLE  -->
    <link href="{{ url_for('static',
filename='css/style.css') }}"  rel="stylesheet" />
     <!-- HTML5 Shiv and Respond.js for IE8 support of
HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the
page via file:// -->
    <!--[if lt IE 9]>
        <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv
.js"></script>
        <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.
min.js"></script>
    <![endif]-->
</head>
<body>
    <header>
    </header>
    <!-- HEADER END-->
    <div class="navbar navbar-inverse set-radius-zero">
        <div class="container">
            <div class="navbar-header">
```

```
                <button type="button" class="navbar-
toggle" data-toggle="collapse" data-target=".navbar-
collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
            </div>

            <div class="left-div">
                <div class="user-settings-wrapper">
                    <ul class="nav">

                        <h4 class="page-head-line"
style="color:white">Docker Based Web Service</h4>

                    </ul>
                </div>
            </div>
        </div>
    </div>
    <!-- LOGO HEADER END-->

    <!-- MENU SECTION END-->
    <div class="content-wrapper">
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    <h4 class="page-head-line">Please
Login or Register To Enter </h4>
                    {% if error %}
                                <p
class=error><strong>Status:</strong> {{ error }}
                        {% endif %}
                </div>

            </div>
            <div class="row">
                <form action="/signup" method="POST"
class="col-md-6">
                  <h4> Register </h4>
                   <br />
                   <label>Enter Email ID : </label>
                        <input type="text" name="username"
class="form-control" />
                        <label>Enter Password :  </label>
                        <input type="password"
name="password" class="form-control" />
                        <hr />
```

```
                        <button  class="btn btn-
warning"><span class="glyphicon glyphicon-user"></span>
 Register </button> 
                    <br /><br />
                     <div class="alert alert-info">
                        You can directly enter if you
already have an acount or register if you don't. Enjoy our
services
                        <br />

                    </div>
                </form>
                <form class="col-md-6" action="/login-gui"
method="POST">
                    <h4> Or Login</h4>
                    <br />
                     <label>Enter Email ID : </label>
                        <input type="text" class="form-
control" name="username"/>
                        <label>Enter Password :  </label>
                        <input type="password"
class="form-control" name="password"/>
                        <hr />
                        <button type="Submit"
value="login" class="btn btn-info"><span class="glyphicon
glyphicon-user"></span>  Log Me In </button> 
                </form>

            </div>
        </div>
    </div>
    <!-- CONTENT-WRAPPER SECTION END-->
    <footer>
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    &copy; 2015 YourCompany | By : <a
href="http://www.designbootstrap.com/"
target="_blank">DesignBootstrap</a>
                </div>

            </div>
        </div>
    </footer>
    <!-- FOOTER SECTION END-->
    <!-- JAVASCRIPT AT THE BOTTOM TO REDUCE THE LOADING
TIME  -->
    <!-- CORE JQUERY SCRIPTS -->
```

```
    <script src="{{ url_for('static', filename='js/jquery-
1.11.1.js') }}" type="text/javascript"></script>
    <!-- BOOTSTRAP SCRIPTS  -->
    <script src="{{ url_for('static',
filename='js/bootstrap.js') }}"
type="text/javascript"></script>
</body>
</html>
```

Kode Sumber 8.2 index-gui.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1, maximum-scale=1" />
    <meta name="description" content="" />
    <meta name="author" content="" />
    <!--[if IE]>
        <meta http-equiv="X-UA-Compatible"
content="IE=edge,chrome=1">
        <![endif]-->
    <title>Docker Based Mininet with SDN
Controller</title>
    <!-- BOOTSTRAP CORE STYLE  -->
    <link href="{{ url_for('static',
filename='css/bootstrap.css') }}" rel="stylesheet" />
    <!-- FONT AWESOME ICONS  -->
    <link href="{{ url_for('static', filename='css/font-
awesome.css') }}"rel="stylesheet" />
    <!-- CUSTOM STYLE  -->
    <link href="{{ url_for('static',
filename='css/style.css') }}" rel="stylesheet" />
     <!-- HTML5 Shiv and Respond.js for IE8 support of
HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the
page via file:// -->
    <!--[if lt IE 9]>
        <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv
.js"></script>
        <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.
min.js"></script>
    <![endif]-->
</head>
<body>
    <header>
```

```
    </header>
    <!-- HEADER END-->
    <div class="navbar navbar-inverse set-radius-zero">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-
toggle" data-toggle="collapse" data-target=".navbar-
collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
            </div>

            <div class="left-div">
                <div class="user-settings-wrapper">
                    <ul class="nav">

                        <h4 class="page-head-line"
style="color:white">Docker Based Web Service</h4>

                    </ul>
                </div>
            </div>
        </div>
    </div>
    <!-- LOGO HEADER END-->
    <section class="menu-section">
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    <div class="navbar-collapse collapse
">
                        <ul id="menu-top" class="nav
navbar-nav navbar-right">
                            <li><a class="menu-top-active"
href="i/ndex-gui">MANAGE</a></li>
                            <li><a href="/add-gui">ADD
DOCKER</a></li>
                            <li><a
href="/formtambahlink">ADD LINK</a></li>
                            <li><a
href="/formlinkupdown">REMOVE LINK</a></li>
                            <li><a href="/logout">LOG
OUT</a></li>


                        </ul>
                    </div>
                </div>
```

```
            </div>
        </div>
    </section>
    <!-- MENU SECTION END-->
    <div class="content-wrapper">
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    <h4 class="page-head-line">MANAGE</h4>

                </div>

            </div>
            <div class="row">
                <div class="col-md-12">
                    <div class="alert alert-success">
                        You can see your switch link here.
                    </div>
                        <a class="btn btn-xs btn-success"
name="seeLink" href="/seelinkofswitch" target="_blank">See
Link</a>
                    <br>
                    <div class="alert alert-success">
                        You can see your docker status and
information here.
                    </div>
                </div>

            </div>
            <br>
            <div class="row">
                 <div class="table-responsive">
                                <table class="table table-
striped table-bordered table-hover">
                                    <thead>
                                        <tr>
                                            <th>ID
DockerHost</th>
                                            <th>Name</th>
                                            <th>Date</th>
                                            <th>Docker
Type</th>
                                            <th>IP
Adress</th>

<th>Status</th>

<th>Action</th>
```

```
                                        </tr>
                                    </thead>
                                    <tbody>
                                        {% for row in rows
%}
                                        <tr>
                                            <td># {{
row[0] }}</td>
                                            <td> {{ row[2]
}}</td>
                                            <td> {{ row[3]
}}</td>
                                            <td> {{ row[5]
}}</td>
                                            <td> {{ row[6]
}}</td>
                                            {% if
row[4]==1 %}
                                                <td>
                                                <label
class="label label-success">UP</label></td>
                                            {% else %}
                                                <td>
                                                <label
class="label label-danger">DOWN</label>
                                                </td>
                                            {% endif %}
                                            <td>
                                                <form
action="/upOrDown" method="POST">

<button class="btn btn-xs btn-warning" name="dockerID"
value="{{row[0]}}">UP/DOWN</button>
                                                </form>
                                                <br>
                                                <form
action="/deleteDocker" method="POST">

<button class="btn btn-xs btn-danger" name="dockerID"
value="{{row[0]}}">DELETE</button>
                                                </form>
                                                <br>
                                                <form
action="/seePort" method="POST" target="_blank">

<button class="btn btn-xs btn-danger" name="dockerID"
value="{{row[0]}}">SEE PORT</button>
                                                </form>
                                            </td>
```

```
                                        </tr>
                                        {% endfor %}
                                        <!--</tr>-->
                                    </tbody>
                                </table>

                            </div>
                        </div>

                        </div>
                    </div>
                </div>
                <!-- CONTENT-WRAPPER SECTION END-->
                <footer>
                    <div class="container">
                        <div class="row">
                            <div class="col-md-12">
                                &copy; 2015 YourCompany | By : <a
href="http://www.designbootstrap.com/"
target="_blank">DesignBootstrap</a>
                            </div>

                        </div>
                    </div>
                </footer>
                <!-- FOOTER SECTION END-->
                <!-- JAVASCRIPT AT THE BOTTOM TO REDUCE THE LOADING
TIME  -->
                <!-- CORE JQUERY SCRIPTS -->
                <script src="{{ url_for('static', filename='js/jquery-
1.11.1.js') }}" type="text/javascript"></script>
                <!-- BOOTSTRAP SCRIPTS  -->
                <script src="{{ url_for('static',
filename='js/bootstrap.js') }}"
type="text/javascript"></script>
</body>
</html>
```

Kode Sumber 8.3 add-gui.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1, maximum-scale=1" />
    <meta name="description" content="" />
```

```
    <meta name="author" content="" />
    <!--[if IE]>
        <meta http-equiv="X-UA-Compatible"
content="IE=edge,chrome=1">
        <![endif]-->
    <title>Docker Based Mininet with SDN
Controller</title>
    <!-- BOOTSTRAP CORE STYLE  -->
   <link href="{{ url_for('static',
filename='css/bootstrap.css') }}" rel="stylesheet" />
    <!-- FONT AWESOME ICONS  -->
    <link href="{{ url_for('static', filename='css/font-
awesome.css') }}"rel="stylesheet" />
    <!-- CUSTOM STYLE  -->
    <link href="{{ url_for('static',
filename='css/style.css') }}" rel="stylesheet" />
     <!-- HTML5 Shiv and Respond.js for IE8 support of
HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the
page via file:// -->
    <!--[if lt IE 9]>
        <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv
.js"></script>
        <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.
min.js"></script>
    <![endif]-->
</head>
<body>
    <header>
    </header>
    <!-- HEADER END-->
    <div class="navbar navbar-inverse set-radius-zero">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-
toggle" data-toggle="collapse" data-target=".navbar-
collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
            </div>

            <div class="left-div">
                <div class="user-settings-wrapper">
                    <ul class="nav">
```

```
                        <h4 class="page-head-line"
style="color:white">Docker Based Web Service</h4>

                    </ul>
                </div>
            </div>
        </div>
    </div>
    <!-- LOGO HEADER END-->
    <section class="menu-section">
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    <div class="navbar-collapse collapse
">
                        <ul id="menu-top" class="nav
navbar-nav navbar-right">
                            <li><a href="/index-
gui">MANAGE</a></li>
                            <li><a class="menu-top-active"
href="/add-gui">ADD DOCKER</a></li>
                            <li><a
href="/formtambahlink">ADD LINK</a></li>
                            <li><a
href="/formlinkupdown">REMOVE LINK</a></li>
                            <li><a href="/logout">LOG
OUT</a></li>

                        </ul>
                    </div>
                </div>

            </div>
        </div>
    </section>
    <!-- MENU SECTION END-->
    <div class="content-wrapper">
        <div class="container">
              <div class="row">
                    <div class="col-md-12">
                        <h1 class="page-head-line">ADD
DOCKER </h1>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-12">
                        <div class="panel panel-default">
                        <div class="panel-heading">
                           SET VARIABLE
```

```
                        </div>
                        <div class="panel-body">
                       <form action="/addDocker"
method="POST">
  <div class="form-group">
    <label for="exampleInputEmail1">DOCKER NAME</label>
    <input type="text" class="form-control"
name="dockerName" placeholder="Enter docker name" />
  </div>
  <div class="form-group">
                                              <label>SELECT
IMAGE</label>
                                              <select
class="form-control" name="dockerImage">
                                                  <option
value="ubuntu:trusty">ubuntu:trusty</option>
                                                  <option
value="mysql/mysql-server:latest">mysql/mysql-
server:latest</option>
                                                  <option
value="nazarpc/phpmyadmin:latest">nazarpc/phpmyadmin:lates
t</option>
                                              </select>
                                          </div>

<br>
<button type="Submit" class="btn btn-success"
value="addDocker">ADD DOCKER</button>

</form>
                            </div>
                            </div>
                    </div>

                </div>
        </div>
    </div>
    <!-- CONTENT-WRAPPER SECTION END-->
    <footer>
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    &copy; 2015 YourCompany | By : <a
href="http://www.designbootstrap.com/"
target="_blank">DesignBootstrap</a>
                </div>

            </div>
        </div>
```

```
    </footer>
    <!-- FOOTER SECTION END-->
    <!-- JAVASCRIPT AT THE BOTTOM TO REDUCE THE LOADING
TIME  -->
    <!-- CORE JQUERY SCRIPTS -->
    <script src="{{ url_for('static', filename='js/jquery-
1.11.1.js') }}" type="text/javascript"></script>
    <!-- BOOTSTRAP SCRIPTS  -->
    <script src="{{ url_for('static',
filename='js/bootstrap.js') }}"
type="text/javascript"></script>
</body>
</html>
```

Kode Sumber 8.4 formlinkupdown.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1, maximum-scale=1" />
    <meta name="description" content="" />
    <meta name="author" content="" />
    <!--[if IE]>
        <meta http-equiv="X-UA-Compatible"
content="IE=edge,chrome=1">
        <![endif]-->
    <title>Docker Based Mininet with SDN
Controller</title>
    <!-- BOOTSTRAP CORE STYLE  -->
   <link href="{{ url_for('static',
filename='css/bootstrap.css') }}" rel="stylesheet" />
    <!-- FONT AWESOME ICONS  -->
    <link href="{{ url_for('static', filename='css/font-
awesome.css') }}"rel="stylesheet" />
    <!-- CUSTOM STYLE  -->
    <link href="{{ url_for('static',
filename='css/style.css') }}" rel="stylesheet" />
     <!-- HTML5 Shiv and Respond.js for IE8 support of
HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the
page via file:// -->
    <!--[if lt IE 9]>
```

```
        <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv
.js"></script>
        <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.
min.js"></script>
    <![endif]-->
</head>
<body>
    <header>
    </header>
    <!-- HEADER END-->
    <div class="navbar navbar-inverse set-radius-zero">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-
toggle" data-toggle="collapse" data-target=".navbar-
collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
            </div>

            <div class="left-div">
                <div class="user-settings-wrapper">
                    <ul class="nav">

                        <h4 class="page-head-line"
style="color:white">Docker Based Web Service</h4>

                    </ul>
                </div>
            </div>
        </div>
    </div>
    <!-- LOGO HEADER END-->
    <section class="menu-section">
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    <div class="navbar-collapse collapse
">
                        <ul id="menu-top" class="nav
navbar-nav navbar-right">
                            <li><a href="/index-
gui">MANAGE</a></li>
                            <li><a href="/add-gui">ADD
DOCKER</a></li>
```

```
                            <li><a
href="/formtambahlink">ADD LINK</a></li>
                            <li><a class="menu-top-active"
href="/formlinkupdown">REMOVE LINK</a></li>
                            <li><a href="/logout">LOG
OUT</a></li>

                        </ul>
                    </div>
                </div>

            </div>
        </div>
    </section>
    <!-- MENU SECTION END-->
    <div class="content-wrapper">
        <div class="container">
              <div class="row">
                    <div class="col-md-12">
                        <h1 class="page-head-line">DELETE
LINK </h1>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-12">
                        <div class="panel panel-default">
                        <div class="panel-heading">
                           SET VARIABLE
                        </div>
                        <div class="panel-body">
                       <form action="/linkupdown"
method="POST">
  <div class="form-group">
    <label for="exampleInputEmail1">Enter Destination
Username</label>
    <input type="text" value="Username of Destination
Switch" class="form-control" name="dest"
placeholder="Enter Destination Username">
  </div>

<br>
<button  class="btn btn-warning">  Delete
Link</button>

</form>
                            </div>
                            </div>
                    </div>
```

```
                </div>
            </div>
        </div>
        <!-- CONTENT-WRAPPER SECTION END-->
        <footer>
            <div class="container">
                <div class="row">
                    <div class="col-md-12">
                        &copy; 2015 YourCompany | By : <a
href="http://www.designbootstrap.com/"
target="_blank">DesignBootstrap</a>
                    </div>

                </div>
            </div>
        </footer>
        <!-- FOOTER SECTION END-->
        <!-- JAVASCRIPT AT THE BOTTOM TO REDUCE THE LOADING
TIME  -->
        <!-- CORE JQUERY SCRIPTS -->
        <script src="{{ url_for('static', filename='js/jquery-
1.11.1.js') }}" type="text/javascript"></script>
        <!-- BOOTSTRAP SCRIPTS  -->
        <script src="{{ url_for('static',
filename='js/bootstrap.js') }}"
type="text/javascript"></script>
</body>
</html>
```

Kode Sumber 8.5 formtambahlink.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1, maximum-scale=1" />
    <meta name="description" content="" />
    <meta name="author" content="" />
    <!--[if IE]>
        <meta http-equiv="X-UA-Compatible"
content="IE=edge,chrome=1">
        <![endif]-->
    <title>Docker Based Mininet with SDN
Controller</title>
    <!-- BOOTSTRAP CORE STYLE  -->
```

```
   <link href="{{ url_for('static',
filename='css/bootstrap.css') }}" rel="stylesheet" />
    <!-- FONT AWESOME ICONS  -->
    <link href="{{ url_for('static', filename='css/font-
awesome.css') }}"rel="stylesheet" />
    <!-- CUSTOM STYLE  -->
    <link href="{{ url_for('static',
filename='css/style.css') }}" rel="stylesheet" />
     <!-- HTML5 Shiv and Respond.js for IE8 support of
HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the
page via file:// -->
    <!--[if lt IE 9]>
        <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv
.js"></script>
        <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.
min.js"></script>
    <![endif]-->
</head>
<body>
    <header>
    </header>
    <!-- HEADER END-->
    <div class="navbar navbar-inverse set-radius-zero">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-
toggle" data-toggle="collapse" data-target=".navbar-
collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
            </div>

            <div class="left-div">
                <div class="user-settings-wrapper">
                    <ul class="nav">

                        <h4 class="page-head-line"
style="color:white">Docker Based Web Service</h4>

                    </ul>
                </div>
            </div>
        </div>
    </div>
    <!-- LOGO HEADER END-->
```

```
    <section class="menu-section">
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    <div class="navbar-collapse collapse
">
                        <ul id="menu-top" class="nav
navbar-nav navbar-right">
                            <li><a href="/index-
gui">MANAGE</a></li>
                            <li><a href="/add-gui">ADD
DOCKER</a></li>
                            <li><a class="menu-top-active"
href="/formtambahlink">ADD LINK</a></li>
                            <li><a
href="/formlinkupdown">REMOVE LINK</a></li>
                            <li><a href="/logout">LOG
OUT</a></li>

                        </ul>
                    </div>
                </div>

            </div>
        </div>
    </section>
    <!-- MENU SECTION END-->
    <div class="content-wrapper">
        <div class="container">
             <div class="row">
                  <div class="col-md-12">
                      <h1 class="page-head-line">ADD
LINK </h1>
                  </div>
                </div>
                <div class="row">
                    <div class="col-md-12">
                        <div class="panel panel-default">
                        <div class="panel-heading">
                           SET VARIABLE
                        </div>
                        <div class="panel-body">
                       <form action="/tambahlink"
method="POST">
  <div class="form-group">
    <label for="exampleInputEmail1">Enter Destination
Username</label>
```

```
    <input type="text" value="Username of Destination
Switch" class="form-control" name="dest"
placeholder="Enter Destination Username">
  </div>

<br>
<button  class="btn btn-success">  ADD LINK</button>

</form>
                            </div>
                            </div>
                    </div>

                </div>
        </div>
    </div>
    <!-- CONTENT-WRAPPER SECTION END-->
    <footer>
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    &copy; 2015 YourCompany | By : <a
href="http://www.designbootstrap.com/"
target="_blank">DesignBootstrap</a>
                </div>

            </div>
        </div>
    </footer>
    <!-- FOOTER SECTION END-->
    <!-- JAVASCRIPT AT THE BOTTOM TO REDUCE THE LOADING
TIME  -->
    <!-- CORE JQUERY SCRIPTS -->
    <script src="{{ url_for('static', filename='js/jquery-
1.11.1.js') }}" type="text/javascript"></script>
    <!-- BOOTSTRAP SCRIPTS  -->
    <script src="{{ url_for('static',
filename='js/bootstrap.js') }}"
type="text/javascript"></script>
</body>
</html>
```

Kode Sumber 8.6 seeLink.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1, maximum-scale=1" />
    <meta name="description" content="" />
    <meta name="author" content="" />
    <!--[if IE]>
        <meta http-equiv="X-UA-Compatible"
content="IE=edge,chrome=1">
        <![endif]-->
    <title>Docker Based Mininet with SDN
Controller</title>
    <!-- BOOTSTRAP CORE STYLE  -->
    <link href="{{ url_for('static',
filename='css/bootstrap.css') }}" rel="stylesheet" />
    <!-- FONT AWESOME ICONS  -->
    <link href="{{ url_for('static', filename='css/font-
awesome.css') }}"rel="stylesheet" />
    <!-- CUSTOM STYLE  -->
    <link href="{{ url_for('static',
filename='css/style.css') }}" rel="stylesheet" />
     <!-- HTML5 Shiv and Respond.js for IE8 support of
HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the
page via file:// -->
    <!--[if lt IE 9]>
        <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv
.js"></script>
        <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.
min.js"></script>
    <![endif]-->
</head>
<body>
<table class="table table-striped table-bordered table-
hover">
    <thead>
        <tr>
            <th>Source</th>
            <th>Destination</th>
        </tr>
    </thead>
    <tbody>
        {% for hasil in hasil %}
        <tr>
            <td> {{ hasil[1] }}</td>
            <td> {{ hasil[3] }}</td>
        </tr>
        {% endfor %}
```

```
        <!--</tr>-->
    </tbody>
</table>
</body>
</html>
```

Kode Sumber 8.7 seePort.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1, maximum-scale=1" />
    <meta name="description" content="" />
    <meta name="author" content="" />
    <!--[if IE]>
        <meta http-equiv="X-UA-Compatible"
content="IE=edge,chrome=1">
        <![endif]-->
    <title>Docker Based Mininet with SDN
Controller</title>
    <!-- BOOTSTRAP CORE STYLE  -->
    <link href="{{ url_for('static',
filename='css/bootstrap.css') }}" rel="stylesheet" />
    <!-- FONT AWESOME ICONS  -->
    <link href="{{ url_for('static', filename='css/font-
awesome.css') }}"rel="stylesheet" />
    <!-- CUSTOM STYLE  -->
    <link href="{{ url_for('static',
filename='css/style.css') }}" rel="stylesheet" />
     <!-- HTML5 Shiv and Respond.js for IE8 support of
HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the
page via file:// -->
    <!--[if lt IE 9]>
        <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv
.js"></script>
        <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.
min.js"></script>
    <![endif]-->
</head>
<body>
    {{ output }}
</body>
</html>
```

8.2 Web API (Flask, Celery, Redis)

Kode Sumber 8.8 tes1.py (Flask API dan Celery)

```
from flask import Flask, session, redirect, url_for,
escape, request, render_template, jsonify, Response
from hashlib import md5
from base64 import b64encode
import MySQLdb
from subprocess import call, Popen, PIPE
from celery import Celery
import fikri
import lordy


app = Flask(__name__)
db = MySQLdb.connect(host="localhost", user="root",
passwd="matapancing", db="tugasakhir")
cur = db.cursor()
p="a"
# Celery configuration
app.config['CELERY_BROKER_URL'] = 'redis://0.0.0.0:6379/0'
app.config['CELERY_RESULT_BACKEND'] =
'redis://0.0.0.0:6379/0'

# Initialize Celery
celery = Celery(app.name,
broker=app.config['CELERY_BROKER_URL'])
celery.conf.update(app.config)


'''
@app.route('/')
def users():
    cur.execute(SELECT * FROM user)
    rv = cur.fetchall()
    return str(rv)
'''

@app.route('/')
def index():
    error = None
    print index
    if 'username' in session:
        return redirect(url_for('indexpage'))
    return render_template('login-gui.html', error=error)
```

```
@app.route('/index-gui')
def indexpage():
    error = None
    print index
    if 'username' in session:
        return redirect(url_for('fetchTable'))
    return render_template('login-gui.html', error=error)

@app.route('/logout')
def logout():
    #this is a comments
    error='Berhasil Log Out!'
    session.pop('username', None) #kalo semua pake clear
    return render_template('login-gui.html', error=error)

@app.route('/fetchTable')
def fetchTable():
    rows = None
    error = None
    hasil = None
    print index
    username_form  = session['username']
    #cur.execute("SELECT COUNT(1) FROM user WHERE username
= %s;", [username form])
    cur.execute("SELECT iduser FROM user WHERE username =
%s;", [username_form])
    hasil=cur.fetchall();
    for hasil1 in hasil:
        idUser_form=str(hasil1[0])

    cur.execute("SELECT * from dockerhost where iduser=
%s;", [idUser_form])
    rows = cur.fetchall()
    #db.commit

    return render_template('index-gui.html', rows=rows,
error=error)

@app.route('/add-gui')
def addgui():
    error = None
    print index
    if 'username' in session:
        return render_template('add-gui.html')
    return render_template('login-gui.html', error=error)

@app.route('/tesnet')
def tesNet():
    return tesNets.delay()
```

```
@app.route('/addswitch')
def lordyAddSwitches():
    source=str(session['username'])
    cur.execute("SELECT iduser FROM user WHERE username =
%s;", [source])
    hasil=cur.fetchall();
    for hasil1 in hasil:
            source_id=str(hasil1[0])
    source_id="s"+str(source_id)
    addSwitchs.delay(source_id)
    return str(source_id)

@app.route('/tesbash')
def tesbash():
    #return str(check_output(["mkdir", "duhdek"]),
shell=True)
    #return str(call(["mkdir",  "duhdek"]))
    output = Popen(['ls', '-a'], stdout=PIPE)
    return output.stdout.read()

#untuk restart jaringan
@app.route('/tescelery')
def tescelery():
    myNetworks.delay()
    return "Network Started"

#Untuk nambah docker
@app.route('/tambahdocker')
def tambahdocker():
    tambahdockers.delay()
    return "docker berhasil ditambah"

@app.route('/formtambahlink')
def formtambahlink():
    return render_template('formtambahlink.html')

@app.route('/formlinkupdown')
def formlinkupdown():
    return render_template('formlinkupdown.html')

@app.route('/seelinkofswitch')
def seelinkofswitch():
    hasil = None
    if 'username' in session:
        username_form=str(session['username'])
        print username_form
        cur.execute("SELECT iduser FROM user WHERE
username = %s;", [username_form])
```

```
        hasil=cur.fetchall();
        for hasil1 in hasil:
            iduser=str(hasil1[0])
        print iduser
        cur.execute("SELECT linkofswitch.source,
userSource.username, linkofswitch.destination,
userDestination.username FROM  `linkofswitch` JOIN  `user`
userSource ON linkofswitch.source = userSource.iduser JOIN
`user` userDestination ON linkofswitch.destination =
userDestination.iduser WHERE source = %s OR destination =
%s", ([iduser], [iduser]))
        hasil=cur.fetchall();
        return render_template('seeLink.html', hasil =
hasil )
    error=None
    return render_template('login-gui.html', error=error)
'''
@app.route('/dockerStop')
def dockerStop():
'''

'''#########################################################
#########################################################
########################################'''

'''Method Get/Post'''

@app.route('/user', methods=['GET', 'POST'])
def user():
    '''if 'username' in session:
        username session = session['username']
        return render_template('index-gui.html',
session_user_name=username_session)'''
    return redirect(url_for('indexpage'))

@app.route('/login-gui', methods=['GET', 'POST'])
def login():
    print login
    error = None
    if 'username' in session:
        return redirect(url_for('indexpage'))
    if request.method == 'POST':
        username_form  = request.form['username']
        password_form  = request.form['password']
        cur.execute("SELECT COUNT(1) FROM user WHERE
username = %s;", [username_form]) # CHECKS IF USERNAME
EXSIST
        if cur.fetchone()[0]:
```

```
            cur.execute("SELECT pwd FROM user WHERE
username = %s;", [username_form]) # FETCH THE HASHED
PASSWORD
            for row in cur.fetchall():
                if password_form == row[0]:
                    session['username'] =
request.form['username']
                    return redirect(url_for('user'))
                else:
                    error = "Salah password!"
        else:
            error = "Anda belum terdaftar!"
    return render_template('login-gui.html', error=error)

@app.route('/signup', methods=['GET', 'POST'])
def signup():
    print signup
    error = None
    if request.method == 'POST':
        username_form  = request.form['username']
        password_form  = request.form['password']
        cur.execute("SELECT COUNT(1) FROM user WHERE
username = %s;", [username_form]) # CHECKS IF USERNAME
EXSIST
        if cur.fetchone()[0]:
            error = "Username sudah digunakan!"
        else:
            cur.execute("INSERT INTO user(username, pwd)
VALUES(%s ,%s)", ([username_form],[password_form]))
            db.commit()
            error = "Berhasil Daftar!"
    return render_template('login-gui.html', error=error)

@app.route('/tambahlink', methods=['GET', 'POST'])
def tambahlink():
    source=session['username']
    dest=request.form['dest']
    status=1
    cur.execute("SELECT iduser FROM user WHERE username =
%s;", [source])
    hasil=cur.fetchall();
    for hasil1 in hasil:
            source_id=str(hasil1[0])
    cur.execute("SELECT iduser FROM user WHERE username =
%s;", [dest])
    hasil=cur.fetchall();
    for hasil1 in hasil:
            dest_id=str(hasil1[0])
    source_id_string="s"+str(source_id)
```

```
    dest_id_string="s"+str(dest_id)
    dest_id=int(dest_id)
    source_id=int(source_id)
    tambahlinks.delay(source_id_string,dest_id_string)
    cur.execute("INSERT INTO linkofswitch(source,
destination) VALUES(%s, %s);", ([source_id],[dest_id]))
    db.commit()
    return redirect(url_for('fetchTable'))

@app.route('/addDocker', methods=['GET', 'POST'])
def addDockers():
    error= None
    print index
    if request.method == 'POST':
        username_form = session['username']
        dockerName_form  = request.form['dockerName']
        dockerImage_form  = request.form['dockerImage']
        #dockerVar form = request.form['dockerVar']

        #cari IDDocker
        cur.execute("SELECT iddocker FROM dockerhost ORDER
BY iddocker DESC LIMIT 1")
        hasil=cur.fetchall();
        for hasil1 in hasil:
            idDocker_form=str(int(str(hasil1[0]))+1)

        #cari UserID
        cur.execute("SELECT iduser FROM user WHERE
username = %s;", [username_form])
        hasil=cur.fetchall();
        for hasil1 in hasil:
            idUser_form=str(hasil1[0])
        #cur.execute("INSERT INTO
dockerhost(iduser,dockername,iddocker)
VALUES(%s,%s,%s)",([idUser form],[dockerName form],[idDock
er form]) )
        ipAddress="10.0.0."+idDocker_form
        cur.execute("INSERT INTO
dockerhost(iduser,dockername,iddocker,jenisdocker,ip_addre
ss)
VALUES(%s,%s,%s,%s,%s)",([idUser_form],[dockerName_form],[
idDocker_form], [dockerImage_form],[ipAddress]) )
        db.commit()

        #Nambah Docker di mininet
        tambahdockers.delay(idDocker_form,
"10.0.0."+idDocker_form+"/8", dockerImage_form,
"s"+idUser_form)
```

```
        return redirect(url_for('fetchTable'))

    #nyeluk shell/mininet script
    return "Under Construction"

@app.route('/upOrDown', methods=['GET', 'POST'])
def upOrDown():
    if request.method=='POST':
        dockerID_form = request.form['dockerID']
        mndocker="mn.d"+dockerID_form
        cur.execute("SELECT dockerstatus FROM dockerhost
WHERE iddocker = %s;", [dockerID_form])
        hasil=cur.fetchall();
        for hasil1 in hasil:
            dockerStatus_form=str(hasil1[0])

        if dockerStatus_form=='1':
            cur.execute("UPDATE dockerhost SET
dockerstatus=0 WHERE iddocker = %s;",[dockerID_form])
            db.commit()
            #popen stop docker
            output = Popen(['docker', 'stop', mndocker],
stdout=PIPE)
            output.communicate()
        else:
            cur.execute("UPDATE dockerhost SET
dockerstatus=1 WHERE iddocker = %s;",[dockerID_form])
            db.commit()
            #popen start docker
            output = Popen(['docker', 'start', mndocker],
stdout=PIPE)
            output.communicate()
        return redirect(url_for('fetchTable'))

@app.route('/deleteDocker', methods=['GET', 'POST'])
def deleteDocker():
    if request.method=='POST':
        dockerID_form = request.form['dockerID']
        mndocker="mn.d"+dockerID_form

        cur.execute("SELECT dockerstatus FROM dockerhost
WHERE iddocker = %s;", [dockerID_form])
        hasil=cur.fetchall();
        for hasil1 in hasil:
            dockerStatus_form=str(hasil1[0])

        if dockerStatus_form=='1':
            output = Popen(['docker', 'stop', mndocker],
stdout=PIPE)
```

```
            output.communicate()
        output = Popen(['docker', 'rm', mndocker],
stdout=PIPE)
        output.communicate()

        cur.execute("DELETE FROM dockerhost WHERE iddocker
= %s;",[dockerID_form])
        db.commit()
        return redirect(url_for('fetchTable'))


@app.route('/linkupdown', methods=['GET', 'POST'])
def linkupdown():
    if request.method=='POST':
        source=session['username']
        dest=request.form['dest']
        linkStatus=9
        cur.execute("SELECT iduser FROM user WHERE
username = %s;", [source])
        hasil=cur.fetchall();
        for hasil1 in hasil:
            source_id=str(hasil1[0])
        cur.execute("SELECT iduser FROM user WHERE
username = %s;", [dest])
        hasil=cur.fetchall();
        for hasil1 in hasil:
                dest_id=str(hasil1[0])
        source_id_string="s"+str(source_id)
        dest_id_string="s"+str(dest_id)
        #select status di link
        #cur.execute("SELECT status FROM linkofswitch
WHERE source = %s AND destination = %s;",
([source_id],[dest_id]))
        #hasil=cur.fetchall();
        #for hasil1 in hasil:
                #linkStatus=str(hasil1[0])
        #kalo up, pencet yg down
        #if linkStatus == '1':
        delLinks.delay(source_id_string,dest_id_string)
        cur.execute("DELETE FROM linkofswitch WHERE source
= %s AND destination = %s;", ([source_id],[dest_id]))
        db.commit()
        #vice versa

            #return "Link Not Found"

#linkUps.delay(source_id_string,dest_id_string)
```

```
            #cur.execute("UPDATE linkofswitch  SET
status=1 WHERE source = %s AND destination = %s;",
([source_id],[dest_id]))
            #db.commit()
        return "berhasil hapusnya"

@app.route('/seePort', methods=['GET', 'POST'])
def seePort():
    if request.method=='POST':
        dockerID_form = request.form['dockerID']
        mndocker="name=mn.d"+dockerID_form

        output = Popen(['docker', 'ps','-f',mndocker, '--
format', '"{{.Ports}}"'], stdout=PIPE)
        output = output.stdout.read()
        #return output
        return render_template('seePort.html', output =
output )
        #output.communicate()
        #return output.stdout.read()

'''#########################################################
#########################################################
########################################'''

'''Celery tasks'''

@celery.task
def myNetworks():
    lordy.myNetwork()

@celery.task
def tambahdockers(idDocker,ipDocker,imageDocker,switch):

lordy.tambahDocker(idDocker,ipDocker,imageDocker,switch)

@celery.task
def tesNets():
    lordy.lordyNet()

@celery.task
def addSwitchs(source_id):
    lordy.lordyAddSwitch(source_id)

@celery.task
def tambahlinks(source,dest):
    lordy.lordyAddLink(source,dest)

@celery.task
```

```
def delLinks(source,dest):
    lordy.lordyDelLink(source,dest)



#Other Method
def searchIDUser(params):
    cur.execute("SELECT iduser FROM user WHERE username =
%s;", [params])
    hasil=cur.fetchall();
    for hasil1 in hasil:
            result=str(hasil1[0])
    return result


app.secret_key = 'awankinton123'
if __name__ == '__main__':
    #tescelery()    #untuk ngestart jaringan
    app.run(debug=False,host= '0.0.0.0')
```

Kode Sumber 8.9 run-redis.sh

```
#!/bin/bash
if [ ! -d redis-stable/src ]; then
    curl -O http://download.redis.io/redis-stable.tar.gz
    tar xvzf redis-stable.tar.gz
    rm redis-stable.tar.gz
fi
cd redis-stable
make
src/redis-server
```

Kode Sumber 8.10 Perintah untuk menjalankan celery

```
sudo celery worker -A tes1.celery --loglevel=info
```

8.3 Virtual Data Center (Containernet)

Kode Sumber 8.11 lordy.py (Kode Containernet)

```
from mininet.net import Containernet
from mininet.node import Controller, RemoteController,
OVSController
from mininet.node import CPULimitedHost, Docker, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

net = 0

def myNetwork():

    global net
    net = Containernet( topo=None,
                   build=False,
                   ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                      controller=RemoteController,
                      ip='0.0.0.0',
                      protocol='tcp',
                      port=6633)

    info( '*** Add switches\n')
    s5 = net.addSwitch('s5', cls=OVSKernelSwitch)
    s4 = net.addSwitch('s4', cls=OVSKernelSwitch)
    s3 = net.addSwitch('s3', cls=OVSKernelSwitch)
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch)

    info( '*** Add hosts\n')
    d4 = net.addDocker('d4', ip='10.0.0.4',
defaultRoute=None, dimage="ubuntu:trusty")
    d6 = net.addDocker('d6', ip='10.0.0.6',
defaultRoute=None, dimage="ubuntu:trusty")
    d1 = net.addDocker('d1', ip='10.0.0.1',
defaultRoute=None, dimage="ubuntu:trusty")
    d3 = net.addDocker('d3', ip='10.0.0.3',
defaultRoute=None, dimage="mysql/mysql-server:latest")
    d2 = net.addDocker('d2', ip='10.0.0.2',
defaultRoute=None, dimage="nazarpc/phpmyadmin:latest")
```

```
    d5 = net.addDocker('d5', ip='10.0.0.5',
defaultRoute=None, dimage="mysql/mysql-server:latest")

    info( '*** Add links\n')
    net.addLink(s1, s2)
    net.addLink(s2, s5)
    net.addLink(s5, s3)
    net.addLink(s3, s1)
    net.addLink(s3, s2)
    #net.addLink(s1, s4)
    net.addLink(s4, s3)
    net.addLink(s4, d1)
    net.addLink(s4, d2)
    net.addLink(s3, d3)
    net.addLink(s3, d4)
    net.addLink(s5, d5)
    net.addLink(s5, d6)

    net.removeLink(None,s4,s3)

    info("*** starting net")
    net.start()


    info("*** Kodingan Tambahan Lordy\n")
    #tambahDocker('11','10.0.0.11/8','ubuntu:trusty','s3')


    #CLI(net)
    #net.stop()

def tambahDocker(nomor,ipAddress,jenis,switchNo):
    global net
    nomor='d'+nomor
    d11 = net.addDocker(nomor, defaultRoute=None,
dimage=jenis)
    net.addLink(d11,net.get(switchNo),params1={"ip":
ipAddress}) #aslinya /8
    #netBuild() unneeded
    #/8 netmask, buat nandain IP jaringannya yg mana, ga
penting se

def netBuild():     #unneeded
    global net
    info( '*** Starting network versi LORDY\n')
    net.build()
    info( '*** Starting controllers\n')
    for controller in net.controllers:
        controller.start()
```

```
    info( '*** Starting switches\n')
    net.get('s5').start([net.get('c0')])
    net.get('s4').start([net.get('c0')])
    net.get('s3').start([net.get('c0')])
    net.get('s1').start([net.get('c0')])
    net.get('s2').start([net.get('c0')])
    info( '*** Post configure switches and hosts\n')

def lordyNet():
    global net
    return str(net.values())
    #return 0

def lordyAddSwitch(switchName):
    global net
    s_user = net.addSwitch(switchName,
cls=OVSKernelSwitch)
    net.get(switchName).start([net.get('c0')])
    #net.addLink(s user,net.get('s1'))

def lordyAddLink(source,dest):
    global net
    net.addLink(net.get(source),net.get(dest))

def lordyDelLink(source,dest):
    global net
    net.removeLink(None,net.get(source),net.get(dest))



if __name__ == '__main__':
    setLogLevel( 'info' )
    #myNetwork()
```

8.4 Testing

Kode sumber yang dilampirkan ini adalah kode sumber untuk pembangunan jaringan virtual untuk keperluan testing pada BAB 5

Kode Sumber 8.12 Pengujian SP-01

```
from mininet.net import Containernet
from mininet.node import Controller, RemoteController,
OVSController
```

```
from mininet.node import CPULimitedHost, Docker, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

net = 0

def myNetwork():

    global net
    net = Containernet( topo=None,
                   build=False,
                   ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                      controller=RemoteController,
                      ip='0.0.0.0',
                      protocol='tcp',
                      port=6633)

    info( '*** Add switches\n')
    net.addSwitch("s1", cls=OVSKernelSwitch)
    net.addSwitch("s2", cls=OVSKernelSwitch)
    net.get("s1").start([net.get('c0')])
    print "..done adding switches"
    print "..adding host"
    d1 = net.addDocker('d1', ip='10.0.0.1',
defaultRoute=None, dimage="ubuntu:trusty")
    d2 = net.addDocker('d2', ip='10.0.0.2',
defaultRoute=None, dimage="ubuntu:trusty")
    net.addLink(net.get("s2"),net.get("s1"))
    net.addLink(d1,net.get("s1"))
    net.addLink(d2,net.get("s2"))


    info("*** starting net")
    net.start()


    info("*** Kodingan Tambahan Lordy\n")

#tambahDocker('11','10.0.0.11/8','ubuntu:trusty','s3')


    CLI(net)
```

```
    net.stop()


if __name__ == '__main__':
    setLogLevel( 'info' )
    myNetwork()
```

Kode Sumber 8.13 Pengujian SP-02

```
from mininet.net import Containernet
from mininet.node import Controller, RemoteController,
OVSController
from mininet.node import CPULimitedHost, Docker, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

net = 0

def myNetwork():

    global net
    net = Containernet( topo=None,
                   build=False,
                   ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                         controller=RemoteController,
                         ip='0.0.0.0',
                         protocol='tcp',
                         port=6633)

    info( '*** Add switches\n')

    for i in range(21):
        if i>0:
            switchName="s"+str(i)
            net.addSwitch(switchName,
cls=OVSKernelSwitch)
            print "Added Switch "+ switchName
            net.get(switchName).start([net.get('c0')])
            print "    "+ switchName+" Connected to
Controllers"
            if i>1:
```

```
net.addLink(net.get(switchName),net.get("s"+str(i-1)))
                print "    "+ switchName+ " Connected
to Previous Switch"

    print "..done adding switches"
    print "..adding host"
    d1 = net.addDocker('d1', ip='10.0.0.1',
defaultRoute=None, dimage="ubuntu:trusty")
    d2 = net.addDocker('d2', ip='10.0.0.2',
defaultRoute=None, dimage="ubuntu:trusty")
    net.addLink(d1,net.get("s1"))
    net.addLink(d2,net.get("s20"))


    info("*** starting net")
    net.start()


    info("*** Kodingan Tambahan Lordy\n")

#tambahDocker('11','10.0.0.11/8','ubuntu:trusty','s3')


    CLI(net)
    net.stop()


if __name__ == '__main__':
    setLogLevel( 'info' )
    myNetwork()
```

Kode Sumber 8.14 Pengujian SP-03

```
from mininet.net import Containernet
from mininet.node import Controller, RemoteController,
OVSController
from mininet.node import CPULimitedHost, Docker, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

net = 0
```

```
def myNetwork():

    global net
    net = Containernet( topo=None,
                    build=False,
                    ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                         controller=RemoteController,
                         ip='0.0.0.0',
                         protocol='tcp',
                         port=6633)

    info( '*** Add switches\n')
    for i in range(101):
        if i>0:
            switchName="s"+str(i)
            net.addSwitch(switchName,
cls=OVSKernelSwitch)
            print "Added Switch "+ switchName
            net.get(switchName).start([net.get('c0')])
            #print "    "+ switchName+" Connected to
Controllers"
            if i>1:

net.addLink(net.get(switchName),net.get("s"+str(i-1)))
                #print "    "+ switchName+ " Connected
to Previous Switch"

    print "..done adding switches"
    print "..adding host"
    d1 = net.addDocker('d1', ip='10.0.0.1',
defaultRoute=None, dimage="ubuntu:trusty")
    d2 = net.addDocker('d2', ip='10.0.0.2',
defaultRoute=None, dimage="ubuntu:trusty")
    net.addLink(d1,net.get("s1"))
    net.addLink(d2,net.get("s100"))


    info("*** starting net")
    net.start()


    info("*** Kodingan Tambahan Lordy\n")

#tambahDocker('11','10.0.0.11/8','ubuntu:trusty','s3')
```

```
    CLI(net)
    net.stop()


if __name__ == '__main__':
    setLogLevel( 'info' )
    myNetwork()
```

Kode Sumber 8.15 Pengujian SP-04

```
from mininet.net import Containernet
from mininet.node import Controller, RemoteController,
OVSController
from mininet.node import CPULimitedHost, Docker, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

net = 0

def myNetwork():

    global net
    net = Containernet( topo=None,
                    build=False,
                    ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                         controller=RemoteController,
                         ip='0.0.0.0',
                         protocol='tcp',
                         port=6633)

    info( '*** Add switches\n')

    for i in range(21):
        if i>0:
            switchName="s"+str(i)
            net.addSwitch(switchName,
cls=OVSKernelSwitch)
            print "Added Switch "+ switchName
```

```
            net.get(switchName).start([net.get('c0')])
            print "    "+ switchName+" Connected to
Controllers"
            if i>1:
                for j in range(i):
                    if j > 0:

net.addLink(net.get(switchName),net.get("s"+str(j)))
                        print "    "+ switchName+ "
Connected to Switch s" + str(j)

    print "..done adding switches"
    print "..adding host"
    d1 = net.addDocker('d1', ip='10.0.0.1',
defaultRoute=None, dimage="ubuntu:trusty")
    d2 = net.addDocker('d2', ip='10.0.0.2',
defaultRoute=None, dimage="ubuntu:trusty")
    net.addLink(d1,net.get("s1"))
    net.addLink(d2,net.get("s20"))


    info("*** starting net")
    net.start()


    info("*** Kodingan Tambahan Lordy\n")

#tambahDocker('11','10.0.0.11/8','ubuntu:trusty','s3')


    CLI(net)
    net.stop()


if __name__ == '__main__':
    setLogLevel( 'info' )
    myNetwork()
```

8.5 Implementasi OpenFlow pada RouterOS dan OpenWRT

8.5.1 Implementasi RouterOS

Untuk mengaktifkan OpenFlow pada RouterOS, lakukan langkah berikut:

1. Buka https://mikrotik.com/download/archive, pada tab all_package-mispbe-6.34.4.zip
2. Buka 192.168.88.1 pada web browser, pilih menu *files*, upload file diatas, lalu reboot
3. Pada 192.168.88.1, pilih menu *interfaces*, Lalu klik port yang ingin dikontrol dengan SDN Controller, buat "Master Port" nya menjadi "None" (Pada percobaan ini, menggunakan port 3 dan 4)
4. Lakukan *telnet* pada *switch* (192.168.88.1), jalankan:

Kode Sumber 8.16 Perintah pada switch MikroTik

```
/interface bridge add name=br1 protocol-mode=none
/interface bridge port add bridge=br1 interface=ether3
/interface bridge port add bridge=br1 interafce=ether4

/openflow add name=fs1 controllers=192.168.202.200
/openflow port add switch=fs1 interface=ether3
/openflow port add switch=fs1 interface=ether4
```

   dengan mengganti "192.168.202.200" menjadi alamat IP dari *host* yang menjalankan SDN Controller
5. Dalam percobaan ini, port 2 digunakan untuk SDN Controller, dan port 5 untuk *virtual data center*, sedangkan port 3 dan 4 untuk port yang disambungkan dengan *host* yang dikontrol oleh SDN Controller.

8.5.2 Implementasi OpenWRT

Untuk memasang OpenWRT pada *switch* MikroTik, dan mengaktifkan OpenFlow, dapat dilakukan dengan menjalankan langkah berikut :

1. Untuk memasang OpenWRT, dapat melakukan langkah langkah seperti pada *paper* berikut[27].

2. Masuk pada *switch* dengan menggunakan *telnet* atau *ssh*.
3. Konfigurasi /etc/config/openflow sebagai berikut:

Kode Sumber 8.17 etc/config/openflow

```
config 'ofswitch'
        option 'dp' 'dp0'
        option 'ofports' 'eth1.11'
        option 'ofctl' 'tcp:192.168.1.217:6633'
        option 'mode'  'outofband'
```

Dengan “192.168.1.217” diganti menjadi IP Adress *host* SDN Controller

4. Konfigurasi /etc/config/network sebagai berikut:

Kode Sumber 8.18 etc/config/network

```
config interface 'loopback'
        option ifname 'lo'
        option proto 'static'
        option ipaddr '127.0.0.1'
        option netmask '255.0.0.0'

config globals 'globals'
        option ula_prefix 'fd5e:72f0:f902::/48'

config interface 'lan'
        option ifname 'eth1'
        option force_link '1'
        option type 'bridge'
        option proto 'static'
        option ipaddr '192.168.1.1'
        option netmask '255.255.255.0'
        option ip6assign '60'

config interface 'wan'
        option ifname 'eth0'
        option proto 'dhcp'

config interface 'wan6'
        option ifname 'eth0'
        option proto 'dhcpv6'

config switch 'eth1'
        option enable_learning '0'
        option reset '1'
        option enable_vlan '1'
        option enable '1'
```

```
config switch_vlan
        option device 'eth1'
        option vlan '11'
        option vid '11'
        option ports '0t 2 3'

config interface
        option ifname 'eth1.11'
        option proto 'static
```

BIODATA PENULIS

Dhanar Prayoga merupakan anak dari pasangan Bapak Bagus Haryosuseno dan Ibu Dina Meinarsari. Ia lahir di Surabaya pada tanggal 27 november 1995. Penulis menempuh pendidikan formal dimulai dari TK Islam Al Azhar Surabaya(1999-2001), SD Islam Al Alzhar Kelapa Gading Surabaya (2001-2007), SMP Negeri 19 Surabaya (2007 – 2010), SMA Negeri 5 Surabaya (2010 – 2013), dan melanjutkan studinya di departemen Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember, Surabaya (2013 – 2017). Bidang studi yang diambil oleh penulis saat berkuliah di Teknik Informatika ITS adalah Komputasi berbasis Jaringan (KBJ). Penulis aktif dalam organisasi seperti Himpunan Mahasiswa Teknik Computer-Informatika(HMTC) sebagai Staf Departemen Hubungan Luar (2014 – 2015) dan sebagai Wakil Ketua Eksternal (2015 – 2016). Penulis Juga Aktif dalam kepanitiaan seperti pada ITS EXPO sebagai Staf Sponsorship (2014) dan Staf Ahli Sponsorship (2015), Schematics sebagai Staf Public Relation (2014) dan Staf Ahli Public Relation (2015). Penulis memiliki hobi membaca buku, bermain, otomotif, dan menyukai hal baru. Penulis dapat dihubungi melalui email dhanarp918@gmail.com